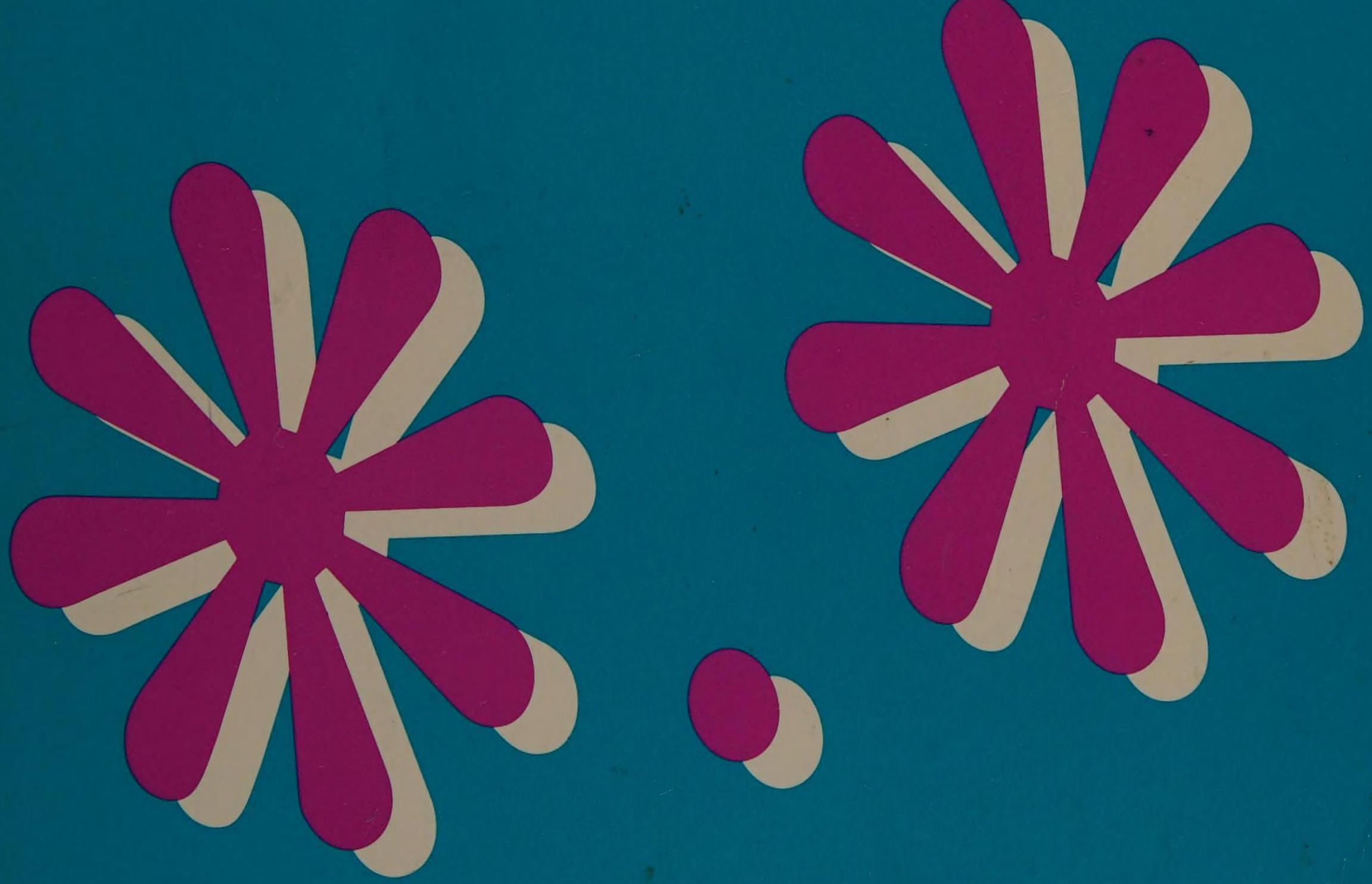
The DOS Coursebook



Forest Lin

The DOS Coursebook

Forest Lin

Tulsa Junior College

Format B: 1510

Dedicated to Dorothy Watson

Scott/Jones Inc., Publishers
P.O. Box 696
El Granada CA 94018
415-726-2436 or 726-7090

ISBN 0-9624230-3-3

Copyright © 1991 by Scott/Jones Inc.
All Rights Reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Printed in the United States of America

Library of Congress Cataloging-in-Publication Number:
QA76.76.063L536 1991
005.265-dc20

Production: Fog Press
Book Manufacturing: Malloy Lithographing

V W X 1 2 3

Contents

Introduction xi
Why This Book xi
What Is in This Book xii
Conventions Used in This Book xii
Version Variances xiii
Suggested Ways of Learning from This Book xiii
Using This Book as a DOS Reference xiv
Using the Included Software xv
About the Author xviii

Chapter One PC and DOS 1
Commands: `DATE, TIME CLS, VER`
What's in a Name 3
Different Kinds of PCs 5
Different Parts of a PC 7
Using the Keyboard 12
System Software and Application Software 15
Starting and Restarting a PC 17
Booting DOS and Entering Date/Time 18
The DOS Prompt and Command Line 20
Ending a DOS Session 24
Summary and Review 26
Review Questions 27

Chapter Two A Quick Tour Through DOS 29

Commands: DIR, TYPE, COPY, REN, DEL (ERASE), DISKCOPY

What Is a File? 32
Changing the Default Drive 34
Displaying Directory Entries 35
Halting a Screen Display 41
Displaying a File's Contents 42
Copying Files 43
Renaming Files 47
Deleting Files 48
Copying a Disk 50
Internal and External Commands 52
Printing a Screen Display and Disk File 53
Summary and Review 57
Review Questions 58

Chapter Three Managing Disks 61

Commands: FORMAT, VOL, LABEL, CHKDSK, FDISK

Different Types of Storage Media 63
Floppy Disk Types and Parts 64
Caring for a Floppy Disk 66
Analysis of Floppy Disks 67
Disk Incompatibility 69
Formatting Floppy Disks 71
What Formatting Does 76
Volume Label 78
Checking a Disk 80
Setting Up a Hard Disk 84
Summary and Review 87
Review Questions 88

Chapter Four Creating and Naming Files 91

Commands: COPY (COPY CON), DIR, TYPE, REN, DEL

Creating Files with COPY CON 93
Appending to a File 94
DOS File Name Conventions 96
Wild Card Characters 101
DOS Editing Keys 106
Creating and Using Batch Files 111
DOS Commands and Batch Files 117
Summary and Review 119
Review Questions 120

Chapter Five Managing Directories 121

Commands: MKDIR (MD), RMDIR (RD), CHDIR (CD), TREE, PROMPT, PATH, APPEND

Organizing a Disk 123
Making and Removing Directories 127
Maneuvering Related Directories 128
DEL (ERASE) and Directories 131
Maneuvering Multilevel Directories 132
Maneuvering Unrelated Directories 134
Maneuvering the Prompt 138
Using TREE to Display Directories 143
Using PATH to Access Executable Files 145
Accessing Data Files with APPEND 146
Summary and Review 150
Review Questions 151

Chapter Six Copying and Comparing Files 155

Commands: COPY, VERIFY, XCOPY, DISKCOPY, COMP, FC, DISKCOMP

The COPY Command's Many Uses 157
Combining (Concatenating) Files 157
Using Wild Cards to Combine Files 159
Changing a File's Date-Time 161
Summary of Combining Files 162
The COPY Command's Switches 165
Copying to Devices 167
Using the XCOPY Command 170
The XCOPY Command's Switches 171
Copying a Whole Disk with DISKCOPY 174
Comparing COPY, XCOPY, and DISKCOPY 175
Copying Copy-Protected Software 176
Comparing Files with COMP 177
Comparing Files with FC 179
Comparing Disks with DISKCOMP 180
Summary and Review 182
Review Questions 183

Chapter Seven Maintaining Files 185

Commands: SYS, REPLACE, ATTRIB, XCOPY, BACKUP, RESTORE, PRINT

Using a New DOS Version 187
Replacing Old Files 190

Setting and Removing File Attributes 193
Backing Up Files with XCOPY 194
Backing Up Files with BACKUP 198
Restoring Backup Files 206
Printing in the Background 209
Summary and Review 213
Review Questions 215

Chapter Eight Redirecting Files 217

Commands: ECHO, MORE, SORT, FIND, FOR..IN..DO

Redirecting Output 220
Redirecting Output with ECHO 222
Redirecting Input 223
Pipes and Filters 227
Using MORE 228
Using SORT 233
Using FIND 238
Combining Filters 241
Inserting Date and Time 243
Summary and Review 247
Review Questions 248

Chapter Nine Configuring the System 251

Commands: MEM, MODE, LASTDRIVE, ASSIGN, JOIN, SUBST, CONFIG.SYS, ANSI.SYS, DEVICE, BREAK, BUFFERS, FILES, SET, VDISK.SYS (RAMDRIVE.SYS), COMSPEC

Reporting Memory Status 253
The Many Uses of MODE 254
Speeding Up the Keyboard 254
Setting Display Modes 256
Setting Printer Modes 257
Reporting Device Status 258
Rerouting Drives and Directories 261
Configuration Commands 267
Installable Device Drivers 270
The DOS Environment 274
The Environment and Batch Files 276
Summary and Review 282
Review Questions 283

Chapter Ten Using Edlin 285

EDLIN Commands: A, C, D, E, I, L, M, P, Q, R, S, T, W

A Quick Tour through EDLIN 288
EDLIN Commands 290
Inserting Lines 290
Editing Lines 294
Moving, Deleting, and Copying Lines 299

Displaying Lines 303
Search and Replace 306
Handling Control Characters 309
Handling Large and Special Files 311
EDLIN and Batch Files 312
Summary and Review 313
Review Quetions 314

Chapter Eleven Programming Batch Files 317

Commands: ECHO, PAUSE, REM, SHIFT, FOR..IN..DO, IF, GOTO, EXIST, CALL

Batch Commands 319
Controlling Screen Display (ECHO, PAUSE, @, REM, :) 320
Maneuvering Dummy Variables (SHIFT, %) 324
Looping (FOR..IN..DO) 327
Branching (IF/IFNOT, GOTO, :, ==) 330
Testing File Existence and Exit Codes (IF EXIST, IF NOT EXIST) 337
Chaining and Nesting (CALL, COMMAND/C) 341
Building a Menu 344
Summary and Review 349
Review Questions 350

Chapter Twelve Using ANSI.SYS 353

Commands: PROMPT, ECHO

ANSI Escape Sequences 355
PROMPT or ECHO 356
Controlling the Cursor 359
Escape Sequences and Batch Files 361
Setting Screen Colors and Modes 365
Redefining Keys 368
Using ANSI.BAT to Set ANSI Values 372
Using MEMO.BAT to Write Memos 374
Using MEMO1.BAT to Write Memos 376
Writing Phone Messages 377
Summary and Review 381
Review Questions 383

Appendix 385
A Glossary 385
B ASCII Characters 399
C DOS Versions 401
D DOS Commands 407
E Installin g and Using the Shell 435
F Elementary Computer Math 441
G Further Reading 443
H Solutions 447

Index 465

Tables

1.1 Different Kinds of PCs 4
1.2 Using Single Keys 14
1.3 Using Combined Keys 14
2.1 Common DOS File Name Extensions 33
3.1 Floppy Disk Configurations 68
3.2 DOS Versions and Disk Capacities 70
3.3 Root Directory Sizes and File Limits 77
4.1 DOS Device Names 98
4.2 DOS Editing Keys 107
5.1 Prompt Metastring Characters 137
8.1 Redirection and Piping Symbols 219
8.2 Directory Entries and Column Positions 235
9.1 Disk Buffers 269
10.1 EDLIN Commands 291
10.2 DOS Editing Keys 295
10.3 Handling Control Characters 307
11.1 Batch Commands 319
11.2 Exit Codes 339
12.1 ANSI Escape Sequences—Cursor and Erasure 359
12.2 ANSI Escape Sequences—Graphics 365
12.3 ANSI Escape Sequences—Key Definition 369
12.4 Single-Number Key Codes 369
12.5 Extended Key Codes 371

Tips & Tricks

Chapter One
Surge Protector 17
Disk Drive Warning Red Light 19
Date and Time 20
Correcting Typing Errors 21
Upper- or Lowercase 22
The PC Calendar 24
Computer Radiation 25
Chapter Two
Quick Formatting 31
README Files 32
What Is a Parameter? 37
What Is a Switch? 39
Drive Failure 40
Paths and Path Names 43
Source and Target 45
DEL and FORMAT 49
Printing Screen Graphics 54
Chapter Three
What Is on a Boot Disk? 73
Recovering a File 84
Chapter Four
File Creation and Existing Files 94
The Syntax of COPY CON 96
A Blank in a File Name 100
The Period as the Current Directory 104
Chapter Five
A Hard Disk's Root Directory 124
Careful Planning for Directories 126
Directory Names 128
Removing a Directory 130
Space versus Backslash 132
Skipping a Directory Path 137
PROMPT $P and a Floppy Drive 141
Handling a Large Tree 145
The Hazards of a TSR Program 149
Chapter Six
Changing File Dates 162
Hazardous COPY Parameters 163
Chapter Seven
Installing Versions 3 and 4 188
Changing from PC-DOS to MS-DOS 189
Protection against Viruses 195

Exit Codes 197
BACKUP's Hazards 204
PRINT Is a TSR 212
Chapter Eight
Error Messages Cannot Be Redirected 222
Printing from the Screen 232
FIND Without Line or File Name 239
Inserting Date-Time Without MORE 244
Chapter Nine
Speedy Cursor 255
The Hazards of Rerouting Drives 266
Expanding the Environment 275
Environmental Cases 277
Chapter Ten
Editing a Backup File 293
EDLIN's Two Buffers 298
EDLIN, ANSI.SYS, and Control Characters 310
Chapter Eleven
ECHO and Empty Lines 322
ECHO and Characters 322
Shifting Parameters 326
The Insensitive Label 331
Parameter Testing 332
Chapter Twelve
Saving the Prompt 358
ECHO On/Off 366
The Ghost of PROMPT 374

Introduction

Why This Book

There are currently some 30 million PCs. Most of these run on DOS. That makes it by far the most popular computer program in the world. This popularity has generated a great deal of demand for DOS books.

There are two kinds of DOS books. One provides enlightenment and instructions for the general public. In this arena, a great variety is available. Most of these books, aimed at "power users," provide comprehensive coverage. They are, however, not particularly suitable for persons starting out.

The other category of books is used in the classroom. Here, the variety is limited. Most DOS textbooks are not terribly sophisticated. They rarely provide comprehensive coverage of the subject. Most of them hold your hand step by step, but never let go of it. This may be necessary as you start out, but this approach becomes a waste of time and precious book space after you learn to walk on your own.

This book aims to combine the best of the above two categories. It will hold your hand and walk you step by step when appropriate, particularly at the beginning. But it goes beyond that. Intricate points are explained in considerable detail and numerous examples are provided. Thus you find in this book a combination of a structured learning environment and in-depth coverage of most topics.

What Is in This Book

There are 12 chapters in this book.[1] Each chapter covers a function or object, or closely related ones. Related DOS commands are explained and many examples provided. Where hands-on experience is required, you are given step by step instructions. Where you need to be aware of important points, the text explains them to you. By combining explanations and concrete steps, we provide the optimal learning environment.

You cannot learn to use a computer program effectively by just reading a textbook; you must roll up your sleeves and get your fingers (and your brain) busy. Learning by doing is the best way of mastering DOS or any other program. To that end, this book is organized to induce you to pause to digest and stop to practice. After one or a few important topics are explained and illustrated, a few Practice exercises and Drill questions will await you. The former gives you an opportunity to practice the steps you have just learned, and the latter lets you reinforce the important points explained in the text.

At the end of each chapter, the salient points are summarized and essay questions are provided. You are to put in writing what you have learned so far. Writing an answer to an essay question requires you to be thoroughly familiar with the material covered in the chapter. This process should further reinforce your learning.

Appendix H provides answers to all the odd-numbered questions in each chapter. For the others, you may need to look up the section(s) preceding the questions to find answers. You can get answers to all the drill questions by running the CAI (computer-aided instruction) program included with the book; see below for details.

Conventions Used in This Book

If you are familiar with computer books, you should have no trouble understanding the simple language used in this book. If you are not, you may need to be aware of the following words and their meanings:

Press	Use a finger to tap a key and let go.
Type	Press keys corresponding to specified letters.
Enter	Type something and end with pressing the Enter key.

For example, you may be instructed to do the following step:

1. Enter the `DIR` command:

```
A>dir
```

Here you are asked to type `DIR` and press Enter. The `A>` notation is automatically displayed by DOS, and your commands are to be entered after it.

[1] A shorter version intended for a short course is being planned. Ask the publisher for details.

In the text, a DOS command name or a file name appears in uppercase (capitalized). This is done for easier reading. With some rare exceptions, you can enter a DOS command or file name in either case. Since you are most likely to enter a command in lowercase, the illustrations of actual commands that you are to enter appear in lowercase.

Learning a new subject requires you to become familiar with its special jargon. To facilitate your familiarization with these terms, a list of New Terms and Symbols appears at the beginning of each chapter. These terms are arranged in the order they appear in the text. They are also boldfaced in the text when they appear for the first time or at crucial places. This arrangement should enable you to find new terms quickly. Appendix A (Glossary) also provides formal definitions for these terms; check it when you find a term you do not understand.

There are dozens of DOS commands. If you want to master DOS, you need to know how to use most of them. The commands to be covered in a chapter are also listed at the beginning of the chapter in the order they are to appear in the text. Some commands are covered in several chapters; they may thus be listed in several chapters.

Version Variances

DOS has gone through four major versions. The latest version, as of this writing, is 4.01 (which is identical to 4.0 except free of bugs). Microsoft, the creator of DOS, continues to market both 3.3 and 4. Many people continue to use previous versions, but 3.3 and 4 are most popular today.

To be as useful to as many people as possible, this book pays attention to both versions 3.3 and 4. Screen illustrations use the latest version, and any significant variance is pointed out in the text (Version Variances). Furthermore, where differences are discussed in the text, a comparison chart appears at the end of the chapter.

As a general rule, a later version includes more commands and features than each version before. A few topics covered in some chapters may not be applicable to you if you use version 3.3. If you use an even earlier version, 3.2, for example, more topics will be irrelevant to you.

If you are not using the latest version, examine Appendix C (DOS Versions) to see what you are missing.

Suggested Ways of Learning from This Book

You may be learning DOS in a short course (a single evening, perhaps), a long course (one semester, three hours per week), or any sort of arrangement in between. Here are some suggestions on how to make the most of this book.

For a short course, try the second half of Chapter 1. Start with Using the Keyboard if you are not already familiar with it. Then work from Starting and

Restarting a PC to the end of the chapter. Then complete Chapter 2. These portions should give you basic knowledge from which you can expand on your own.

The book has enough material for a course that is a semester long meeting three times a week. You can follow the chapter sequence and do all the exercises. Or you can jump from Chapter 4 to Chapter 10, and then back to Chapter 5 and the remaining chapters. This will introduce to you the line editor (EDLIN) at an earlier stage and enable you to use it to write batch files sooner. Using EDLIN to write batch files is much more convenient than using `COPY CON` (discussed in Chapter 4). If you do not intend to write many batch files, however, you are not likely to have much use for this editor.

Using This Book as a DOS Reference

There are numerous DOS commands, and the number has been increasing with each new version. After you learn some new commands, you are likely to forget some of the old commands you learned earlier. You may then need to look up earlier information.

This book is designed with easy reference in mind. You can use a number of features to quickly locate any significant information in the text. Some of these are placed at the end of the book, including:

- **Index**
- **Glossary**
- **DOS Commands**

You can quickly find information by looking up relevant terms in the index, which is quite comprehensive.

As mentioned earlier, the beginning of each chapter lists terms and DOS commands to be discussed in the chapter. Furthermore, major points and command examples are listed under the title of Quick Start. These items may not mean much to you as you start out. But in the future, when you want to come back to find something you have forgotten, you can use them to quickly locate more related information in the chapter.

Tables and Tips & Tricks are scattered throughout the book. These are indexed in the table of contents. Some tables listing commands or options are most useful as quick reminders.

The quickest way to remind yourself how to use a command or what switches are available with it is to look in Appendix D (DOS Commands). It provides the most vital information and examples for most DOS commands. If you want even quicker access to the information in it, try `DOS.EXE`, explained below.

Using the Included Software

This book comes with a floppy disk, which is labeled DOS-TA (for DOS Teaching Assistant) and contains the following items:

1. `DOS.EXE`—online help program for DOS commands
2. `DRILL.EXE`—CAI program
3. `DOSLESnn`—question/answer files (e.g., `DOSLES3` for Chapter 3) to be used by the CAI program
4. `MENU.EXE` & `MENUB.BAT/KEY.COM`—menus for the CAI program and the question/answer files
5. `SAMPLE.TXT`—a practice text file used in the book

To get a screen of online help for a DOS command, follow these steps:

1. Insert DOS-TA in drive A.
2. Enter a command like this:

```
A>dos copy
```

This command will display a screen of information related to the `COPY` command. If you enter the program's name alone, like this:

```
A>dos
```

the screen will display all the available options. If you enter a command that is not available, a similar screen will also appear.

Each screen usually displays information related to a DOS command. All the available switches are listed and plenty of examples are provided. You can thus quickly find out how to use a command or an available switch. If you need more details, you can look up the chapter(s) pointed out on each screen.

A few screens show information not specifically related to individual DOS commands. These include KEYS (special keys used by DOS), SYMBOLS (special symbols recognized by DOS), BATCH (general information related to batch files), EDLIN (a summary of EDLIN commands) and ANSI (a brief explanation of ANSI.SYS).

The `DOS.EXE` program resembles what you see in Appendix D. You can use either as a quick reference.

The `DRILL.EXE` program provides a CAI interactive mechanism for the drill questions appearing in each chapter of the book. To run a lesson, enter a command like this:

```
A>drill dosles1
```

This will start the drill questions for Chapter 1. If you enter the program's name alone, like this:

```
A>drill
```

you will be asked to enter a lesson after the program is started.

If you start a drill lesson with either method above, you can store the program and the lesson files in different drives. If the program is in drive A and a lesson file in B, you can enter a command like this:

```
A>drill b:dosles3
```

Once a drill lesson begins, you are given a question and prompted to supply an answer, which could be a letter, number, word, or phrase. The program will try to match your answers to those stored in the lesson file. You can give an answer in uppercase or lowercase.

A running scoreboard is displayed at the bottom of the screen. It keeps track of how many right and wrong answers you have given. If you give a right answer, some acclamation is indicated. If you supply a wrong answer, a brief explanation is displayed and you are prompted to press a key to go to the next question.

Each lesson contains 30 questions. After you finish all of them, the final score will be displayed. If you want to run the same lesson or another one, just repeat the above steps. If you want to abort a lesson, press Ctrl-C (or Ctrl-Break) at any time.

If you are a beginner, you may want to take advantage of two menu programs to simplify your access to a drill lesson. `MENU.EXE` is a program through which you can access any of the drill lessons. To run the program, just enter this:

```
A>menu
```

When the menu screen appears, you can just press a key to select any of the available options displayed on the screen.

`MENUB.BAT` is a batch file comparable to the menu program explained above; `KEY.COM` provides interactive mechanism for the batch file. It runs rather slowly, as all complex batch files do. However, you can customize it and use it as an example for creating a similar menu batch file.

If you use either menu to start a drill lesson, all the files must be on the same disk and directory. If you start manually, as explained above, the files can be in different drives.

Acknowledgments

Good books are the result of effective collaboration between the author and many other individuals. I have been fortunate to have had a variety of individuals help make this a better textbook. The following individuals provided me with many helpful and insightful comments over several drafts of the manuscript:

Carl Grame and Dan O'Donnell
De Anza College

Jerry Martin
Ohio State University

Ivan Wallace
East Carolina University

Donna Benevich
Foothill College

Jane Varner
Montgomery College

Bill Barry
El Paso Community College

Margaret Randall
Canada College

Mary Rasley
Lehigh County Community College

My editor and publisher Richard Jones made many suggestions regarding possible improvements during the successive drafts of this manuscript. Although I have not always followed his advice, I have always appreciated the fact that he was paying attention to details, and was always focused on the goal of making a better book.

The copyediting by Sheryl Rose added substantially to the clarity of the exposition and organization of this book. The proofreading by Rene Lynch on the page proof for this work caught and corrected many errors, typos, and oversights which even students did not find. The typesetting and design by Fog Press was first rate. Without the care and attention to detail exhibited by these individuals, the overall quality of this book would not be as high as it is.

Class Testing

All of us know that we don't really know whether our texts are effective until we've tried to use them in our classes. In an effort to take every step towards developing the best possible book, a preliminary edition of this text was used by Dorothy Calvin at Skyline College. The final product benefited greatly from the comments of several classes of students at Skyline College, as well as from the constructive and astute criticisms of Professor Calvin based on her teaching experience.

About the Author

Born in Taiwan, the author has lived most of his life in America. He holds a B.A. in accounting and a B.A., M.A., and Ph.D. in political science. He has taught political science and computer-related courses in various colleges for 20 years. He currently teaches at Tulsa Junior College.

Dr. Lin has published three computer books and contributed chapters to other books; one of the books (*WordPerfect Macro Library*, Que, 1988) was featured in a monthly selection of the Small Computer Book Club. His involvement with the PC and DOS began when the IBM PC was first introduced. He has learned to use several computer languages and scores of software packages. He used the C programming language to write the programs explained above.

If you find inaccuracies, ambiguities, inconsistencies, inadequacies and other deficiencies in this book or the included software, or have comments and suggestions, please write to:

Dr. Forest Lin
Tulsa Junior College
909 S. Boston
Tulsa, OK 74119

or to

Scott/Jones, Inc.
P.O. Box 696
El Granada, CA 94018
(415) 726-2436
(415) 726-7090

PC and DOS

Chapter 1

TOPICS:

What's in a Name
Different Kinds of PCs
Different Parts of a PC
Using the Keyboard
System Software and Application Software
Starting and Restarting a PC
Booting DOS and Entering Date/Time
The DOS Prompt and Command Line
Ending a DOS Session

COMMANDS COVERED:

`DATE, TIME, CLS, VER`

NEW TERMS AND SYMBOLS:

PC, DOS, mainframe, mini, micro, KB, RAM, XT, AT, CPU, clone, compatible, PS/2, PS/1, MCA, EISA, bus, system unit, peripheral, motherboard, ROM, firmware, BIOS, boot, expansion slot, controller card, video graphics card, MDA, pixels, Hercules, RGB, CGA, EGA, VGA, parallel port, daisy-wheel printer, dot-matrix printer, NLQ, laser printer, mouse, serial port, modem, baud, scanner, cursor, toggle, enhanced keyboard, system software (operating system), application software, CP/M, PC-DOS, MS-DOS, OS/2, UNIX, operating environment, surge protector, boot (system) disk, DOS (system) prompt, DOS command, command line

REQUIRED MATERIALS:

1. A basic PC system
2. A DOS boot (system) disk—if your PC is not equipped with a hard disk

Quick Start

✎ **To distinguish different kinds of PCs:**
See Table 1.1.

✎ **To use some of the special keys and combinations on the keyboard:**
See Table 1.2 and Table 1.3.

✎ **To boot (start) DOS:**
Insert a system disk in drive A and turn on the power. If you use a hard disk, leave the door of drive A open and turn on the system unit.

✎ **To restart DOS:**
Press the Reset button (if any) on the system unit or press this key combination on the keyboard: Ctrl-Alt-Del.

✎ **To clear the screen:**

```
A>cls
```

✎ **To show the DOS version you are using:**

```
A>ver
```

✎ **To show and/or change the current date:**

```
A>date
```

✎ **To show and/or change the current time:**

```
A>time
```

✎ **To end a DOS session:**
Turn off the system unit, printer, and monitor; retrieve the system disk from drive A, if any.

Lots of new terms, many of them acronyms and abbreviations, will greet you in this chapter. If you are versed in PC (Personal Computer) technology, these should be your household words. If you are a newcomer (or have not kept up with the latest developments), you need to be familiar with this new jargon. The terms explained in this chapter will enable you to understand and use the PC and **DOS** (Disk Operating System).

PC technology changes rapidly. New terms and acronyms sprout up every week. If you want to keep up with the latest developments, the Further Reading list in Appendix G should be helpful.

This chapter introduces some commonly used PC hardware items, some software terms, and a few simple DOS commands. They will get you started on your way to mastering the PC and DOS.

What's in a Name

✎ **Computers can be grouped according to their sizes (from large to small) into mainframes, minis, and PCs. PCs in turn can be divided into XT, AT, PS/2, PS/1 and others.**

Computers used to be large, centrally located machines that served a lot of people. Today these mainframe computers, or mainframes for short, still exist, but there are other computers as well. Smaller-scale computers, serving fewer people and having less power, are called minicomputers, or **minis** for short. Next come personal computers.

When they first appeared in the mid-1970s, tiny computers were called microcomputers, or **micros** for short. The first wave of micros, for games or business uses, generally had limited capabilities.

IBM (International Business Machines) introduced its first micros in 1981, using the term **PC**. Many people began to apply this term to a computer that was supposed to do real work, not play. Over the years, the two terms have become blurred, and PC has mostly supplanted micro. Today, the term PC is used as a generic name referring to all small computers below minis and above game machines.

The initial IBM PC was a simple machine. It had 16**KB** of **RAM**. (RAM stands for random access memory, or simply computer memory; KB stands for kilobyte, or 1000 bytes. A byte consists of 8 bits. Appendix F explains bytes, bits, and other units of measurement in computer math.) The PC also came with one or two floppy disk drives. This arrangement soon proved to be inadequate. In 1983 IBM introduced a new model equipped with up to 256KB of RAM and a hard disk (also known as a Winchester disk; IBM calls it a fixed disk because it is not supposed to be removed like a floppy disk). This model was named *XT* (Extended Technology). In 1984, IBM introduced a new computer based on a new **CPU** (central processing unit, a PC's "brain" where all the calculation is done), named **AT** (Advanced Technology). Although the original PC and XT are in the same class, the AT is a much more powerful machine.

Success breeds imitation and competition. The phenomenal popularity of the PC led to a tidal wave of imitators known as IBM **clones** (exact copies) or **compatibles** (work-alikes). They can run the programs written for IBM products or accept parts and devices intended for genuine IBM machines.

Compatibles vastly expanded the PC market. Most of them cost less than half of what IBM was charging. Lower prices made PCs affordable to an increasing number of people, but drastically reduced IBM's market share.

In 1987 IBM abandoned the AT in favor of the **PS/2** (Personal System/2). The PS/2 consists of a variety of products based on different Intel CPU chips (explained in the next section). Most of the models are equipped with a new device, an **MCA** (Micro Channel Architecture) **bus** (a circuit that carries data and allows different computer parts to communicate with one another). The MCA is protected by a wall of legal patents, which make it difficult to clone legally and technically.

IBM brought back the AT in mid-1990 by introducing a new line of computers under the label of PS/1. Like the original AT, the PS/1 features the 286 CPU and the traditional AT (not MCA) bus. The introduction of this new line marked IBM's second entry to the home market. These real computers were much better received than the toy-like PCjr, IBM's first product aimed at the home market.

Table 1.1 Different Kinds of PCs

Name	Meaning
PC	Personal Computer, the original IBM PC; generic term for all small-scale computers
XT	Extended Technology, the original IBM PC/XT; generic term for a PC based on the 8088 or compatible CPU
AT	Advanced Technology, the original IBM PC/AT; generic term for a PC based on the 80286 or higher CPU
PS/2	Personal System/2, the new line of PCs marketed by IBM; contains various models based on different CPUs; some have an MCA bus, others have the traditional XT or AT bus
PS/1	The PCs intended for the home market. All models contain the 286 CPU and the traditional AT bus

Reflecting the new market reality, the term PS/2 (or PS/1) has become part of PC terminology. Today, IBM's share of the PC market is about one-third. The rest is controlled by other companies such as Compaq and Tandy. Much of the world still refers to this category of computers as PC, in which the PS/2 (or PS/1) is a subcategory of the PC.

As the 1990s dawned, **EISA** (Extended Industry Standard Architecture) has become another popular PC acronym. It is an improved AT **bus** sponsored by a

consortium of IBM rivals to counter IBM's proprietary MCA bus. IBM claims that its MCA is more efficient than the old AT bus. EISA members maintain that EISA is just as good as MCA and, unlike MCA, will accept the numerous existing AT controller cards (thus allowing users to save money). The battle of the buses promises to be quite a spectacle in the new decade; its outcome may determine which vehicle will carry us to the next century.

Drill

Select an acronym in the list to answer questions 1–6:

a. KB, b. CPU, c. EISA, d. MCA, e. RAM, f. PS/2.

_____ 1. Computer memory.

_____ 2. A line of PCs marketed by IBM.

_____ 3. The "brain" of a PC.

_____ 4. A unit of measurement.

_____ 5. IBM's new bus structure.

_____ 6. An extension of the AT bus.

Different Kinds of PCs

✎ **Most PCs are based on a variety of CPUs made by Intel: 8088/8086, 80286, 80386, and 80486.**

There are different ways to classify PCs. It can be done by brand, such as IBM, Compaq, Tandy, Epson, etc. It can be done by size, such as tower (upright, floor-mounted), desktop (sitting on top of a desk), portable (also known as luggable), laptop (can be held on a person's lap), notebook (about the size of a hardbound book), and pocket (can be put in a pocket). The most meaningful classification, however, is by the CPU a PC uses.

Most PCs use CPU chips made by Intel Corporation, including the 8088/8086, 80286, 80386, and 80486. If you intend to buy a PC, make sure you know what chip you are getting. Generally speaking, a PC based on a higher-numbered chip costs more and runs much faster.

8088/8086

The first IBM PC and later the IBM XT were based on the 8088 chip, operating at 4.77 MHz (megahertz, or 1 million cycles per second) speed. The new PS/2 line also

includes at the low end some models (Models 20–30) using a slightly improved 8086 chip operating at 8 MHz.

Today, many vendors also market entry-level PCs based on the 8088 chip running at 10 MHz. Some also use a clone chip named V20 manufactured by the Japanese NEC Corporation.[1]

80286

When IBM introduced the AT in 1984, the new model was based on the 286 chip running at 6 MHz. This was a vast improvement over the previous models. This model can run two or three times faster than the old ones.

AT clones continue to become faster. Compatible models running at 8 MHz were introduced shortly afterward. Then came 10, 12, and (as of this writing) 25 MHz. Failing to stem the tide of clones and rapidly losing market share to competitors, IBM joined the march toward higher speeds, and finally gave up the entire AT line in favor of the PS/2 in 1987.

The 286 market continues to thrive. As of 1990, the 286-based PCs outnumber those based on other chips. Some mid-level PS/2 models (Models 40–50) and the entire PS/1 line are also based on this chip. This chip is no longer manufactured by Intel, but by rivals who got licenses from Intel. Intel now concentrates on the more lucrative 386 and 486 chips.

Today, a person doing complex tasks needs a PC that is based on the 286 or a higher-numbered chip. Most new operating systems such as OS/2 and UNIX (explained below) can run only on a PC with a 286 or higher CPU. An increasing number of application programs are leaving behind the slower 8088/8086 chips in favor of the speedier and more capable ones.[2]

80386 and Beyond

Introduced in 1986, the 386 chip (and the less capable 386SX, which is a hybrid of 286 and 386) has been growing faster in sales than the lower-end ones. Initially offered at 16 MHz, then 20, then 25, and finally 33, this chip can vastly speed up a PC. A 386-based PC can now rival in performance a minicomputer that costs ten times as much. As of mid-1990, a reasonably equipped single-user 386 PC costs about $2,000. It is about half of what an initial 8088-based IBM PC would cost but has a tenfold increase in performance. Using this measurement, today's $1 can purchase 20 times the computing power of less than a decade ago.

An increasing number of operating systems and application programs are written specifically for the 386. They can run only on PCs using this chip. In time, this chip is expected to become the mainstay of the PC world, and those earlier ones will be used for simpler and slower tasks.

[1] Intel filed a lawsuit against NEC for copyright infringement. NEC won the case. Today NEC continues to market the V20 and its improved version, the V30 (a clone of the 8086 chip).

[2] For example, spreadsheet programs such as Microsoft Excel and Lotus 1-2-3 release 3 can no longer run on a PC with an 8088/8086 chip. If you have an 8088-based PC, you must use Lotus 1-2-3 release 2.2, not release 3.

In 1990, more powerful PCs based on the 486 CPU began to appear, mostly in multiuser environments. These PCs can run programs written for the 386 and be used by dozens of people at the same time. Corporations like them because they can cut the cost of computing. They are expected to replace many minis and mainframes.

Others

There are currently other small-scale computers that are quite capable and powerful. These include the Apple Macintosh and various models produced by Atari, Commodore, NeXT, Sun, and others. Most of these are based on the 68000 line of CPUs produced by Motorola.

Although they can rival the most powerful PCs based on the Intel CPUs, these computers mostly are not compatible with IBM-standard models and cannot run software packages intended for IBM or IBM clones. Most of them also use an operating system different from and not compatible with DOS.

Different Parts of a PC

✎ **A PC system is comprised of a system unit (which contains a motherboard, CPU, ROM, RAM, etc.), one or more disk drives, a monitor, a keyboard, a printer, and other peripherals.**

A PC system typically consists of a **system unit,** one or two disk drives (usually installed inside the system unit), a monitor, a keyboard, and a printer. These are the bare minimum for a complete system. For more exotic applications, a PC could be connected to other **peripherals** like a mouse, modem, scanner, etc.

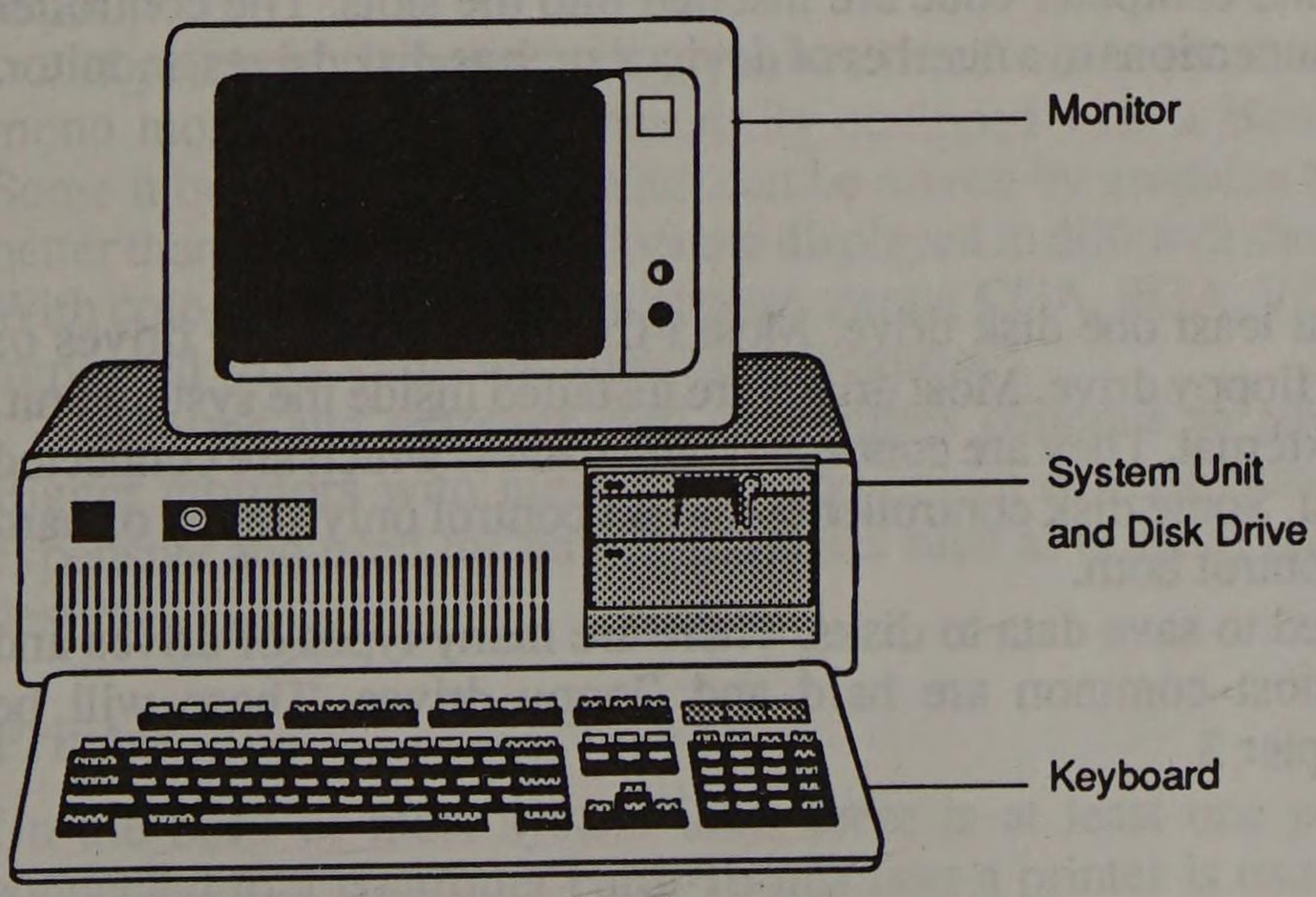


Figure 1.1 A personal computer

System Unit

Also referred to as the box, case, or chassis, the system unit is a rectangular metal box whose size varies greatly. It contains the vital components of a PC.

At the bottom, inside a system unit, rests a **motherboard.** It serves as the base for connecting all the parts, including the CPU, ROM, RAM, and expansion slots.

The CPU is the brain of a PC. As explained in the preceding section, it determines almost everything else in a PC.

ROM (read-only memory) is an example of **firmware,** which lies between hardware and software. It is a hardware part containing software program code. The code is permanently etched on the memory and stays there with or without power. When power is turned on, the code is read and executed; when power is off, it stays dormant.

An important component of a PC's ROM is a **BIOS** (basic input-output system) chip. It provides the initial instructions to the PC.[3] After that, DOS is **booted** (started) from the system files stored on disk.

Some laptop and pocket PCs have DOS and other programs stored in ROM. When power is turned on, DOS is automatically booted from ROM. This saves power and disk space, but reduces a machine's versatility. A newer DOS version, for example, cannot be run by such a PC.

A PC needs working space to maneuver computer code and data. This working space is **RAM** (random access memory), also called memory or dynamic memory. RAM is volatile. When power is on, it comes to life; when power is off, it is dead. Any computer code or data in RAM will disappear when power is off.

When you buy a PC, you need to know how much memory comes with it. Some 8088-based PCs are marketed today with as little as 256KB of RAM. Most application programs, however, need 512KB and a few need 640KB. At the present time, DOS can use up to 640KB of RAM.

To provide an interface with an outside device, the motherboard has a number of **expansion slots. Controller cards** (also known as adapters, boards, or cartridges) with special circuitry and computer code are inserted into the slots. The controller cards in turn provide connections to a number of devices such as disk drives, monitor, printer, etc.

Disk Drives

Each PC has to have at least one disk drive. Most PCs have two floppy drives or one hard drive and one floppy drive. Most drives are installed inside the system unit, although they can be external. They are connected with cables, which are connected to a disk controller card. Some disk controller cards can control only floppy or hard disks, but others can control both.

Disk drives are used to save data to disks. There are many types of drives and storage media. The most common are hard and floppy drives. These will be explained fully in Chapter 3.

[3] Old IBM models also store a simplified BASIC language in ROM. If the BIOS cannot find DOS to boot, this BASIC is loaded. In that case, you cannot use a disk drive.

Monitor

A monitor is used to display information entered from the keyboard or read from a disk. It is connected to the system unit via a jack in the back; the jack is in turn connected to a **video graphics card.** There are many kinds of monitors, each of which has to be matched with a compatible video graphics card.

When the first IBM PC was introduced, buyers could choose a monochrome (one-color) monitor or a color monitor. The mono monitor was matched with an **MDA** (Monochrome Display Adapter). It could display only characters, but not graphics, which consist of a series of screen **pixels** (dots).

The **Hercules** graphics card (HGC) was soon introduced to let a monochrome monitor display both text and graphics. Besides displaying crisp characters (each formed in a 9 x 14 pixel matrix), it can provide a high-resolution graphics display of 720 (horizontal) by 348 (vertical) pixels. This proved to be wildly popular and widely imitated. Today, most mono monitors are equipped with this monographics card (MGA), and most graphics programs can also utilize its capabilities.

IBM's first color monitor is called **RGB** (Red Green Blue) and is paired with the **CGA** (Color Graphics Adapter) card. This combination produces poorly formed characters (8 x 8) and coarse graphics of 320 x 200 screen pixels in medium-resolution (color) mode and 640 x 200 in high-resolution (black and white) mode. Only 4 colors can be displayed at one time out of a palette (possible choice) of 16 colors.

The AT brought with it the **EGA** (Enhanced Graphics Adapter) and its matching monitor. Characters become fuller (8 x 14) and screen pixels are increased to 640 x 350 (new ones, not produced by IBM, go up to 640 x 480). A total of 16 colors out of a palette of 64 can be simultaneously displayed.

The PS/2 also appeared with a new video standard—the **VGA** (Video Graphics Array) card and monitor pair. Characters appear in 9 x 16 matrixes, screen graphics pixels are increased to 640 x 480 pixels, and 256 colors can be simultaneously displayed out of a palette of 256K (256 x 1024). (Low-end PS/2 models are equipped with MCGA, or Multicolor Graphics Array, which is a slimmed-down version of VGA.)

If you buy a monitor today, you are faced with a confusing array of choices. A mono monitor is most economically equipped with a Hercules-compatible card. Some mono monitors, however, can be driven by graphics cards comparable to or better than EGA or VGA; colors are displayed in different shades of gray (or brown). With color monitors, you can choose among CGA, EGA, VGA, improved EGA, or improved VGA cards and matching monitors.

Monitor and graphics-card capabilities continue to advance at a rapid pace. Bigger monitors with more pixels and colors are introduced regularly. They are expensive and used for limited purposes such as desktop publishing and industrial design.

Printer

On the back of most system units there is at least one **parallel** port, which is connected to a controller card. To this port a printer is usually connected. (Some printers are connected to serial ports.)

Various kinds of printers are available for a PC. In the beginning, the **daisy-wheel printer** was popular, highly prized, and highly priced. Like a typewriter, it can print whole characters. Although its print quality is excellent, it is slow and not very versatile. Printing characters of a different size or style requires changing the daisy wheel, or print element. Printing graphics is impossible. These printers are rapidly being phased out.

The **dot-matrix printer** prints characters by putting together dots in the form of a matrix. In the beginning, 7 pins (wires) were used to form crude-looking characters in a 7 x 7 dot matrix. Dot-matrix technology advanced rapidly. Soon, 9-pin models appeared; these inexpensive machines are still very popular today. At the high end, 24-pin models can create **NLQ** (nearly letter-quality) characters with a matrix of 24 × 36 dots.[4] Graphics can also be printed in 360 x 360 dots per inch (DPI). These printers operate at a much higher speed than daisy wheels. With improved print quality and continuous advance in speed, plus rapidly falling prices, they have become the most popular type of PC printers.

The **laser printer** is the aristocrat of the printer world. Printing at 8 or more pages per minute, they are much speedier than other printers. They print good-looking text and graphics at a relatively high resolution of 300 x 300 dots per inch; professional typesetting machines go beyond 1000 x 1000 DPI. Their most attractive attribute is probably the ability to handle different font sizes and styles. A newsletter heading, for example, can be printed in large characters of a certain style while the text is in smaller characters of another style.

High prices inhibited the laser printer's widespread use. But in late 1989, Hewlett-Packard Company, the leader in this increasingly competitive market, began to market a relatively slow 4-page-per-minute model named LaserJet Series IIP at a retail price below $1000, which is less than the price of some of the high-end dot-matrix printers. If this trend continues, laser printers may eventually replace dot matrixes as standard PC printers.

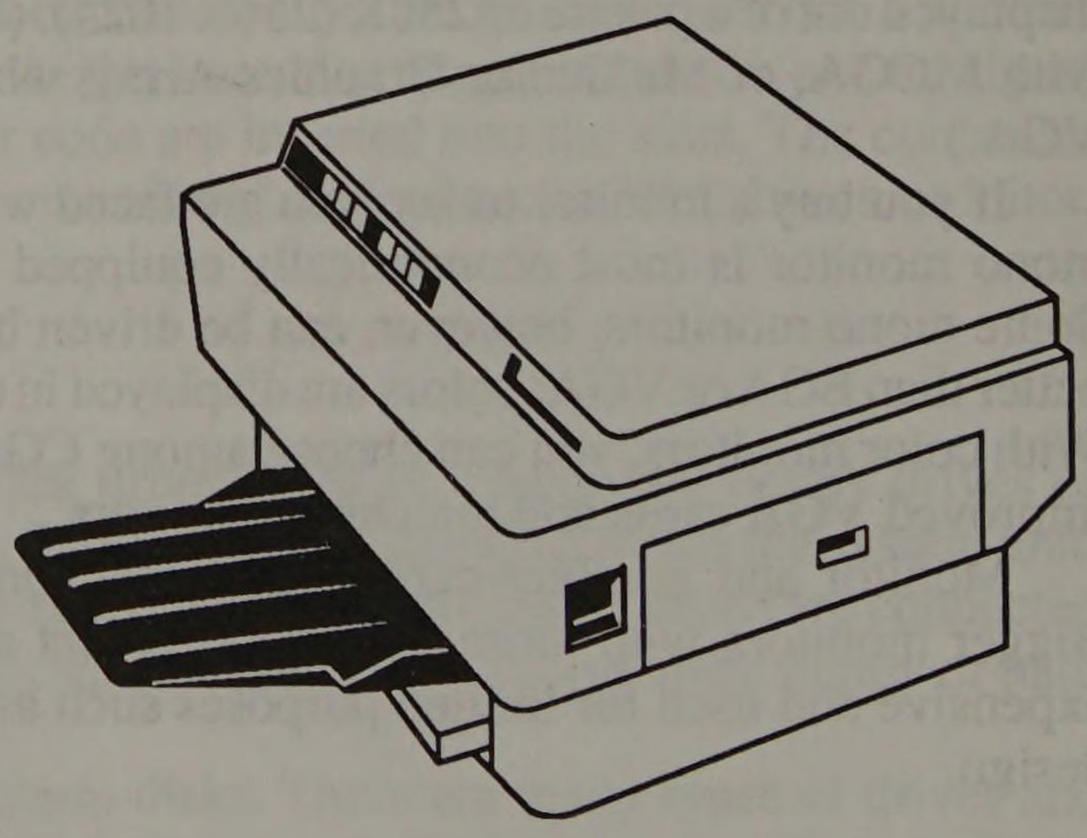


Figure 1.2 A laser printer

Other Peripherals

A host of other peripheral devices can be connected to a PC. The most common are the mouse, modem, and scanner.

A **mouse** is an input device that can be connected to an expansion slot (bus mouse) or a **serial port** (serial mouse). A mouse can maneuver menus and graphics objects. If you

[4] The 24-pin dot-matrix technology was intitially developed by Japanese companies to handle their complicated kanji (Chinese) characters.

use DOS version 4, you can use a mouse to move among various menu items and enter a command by clicking a button. The most useful task for a mouse, however, is handling graphics. Some graphics software packages are difficult or impossible to handle without using a mouse.

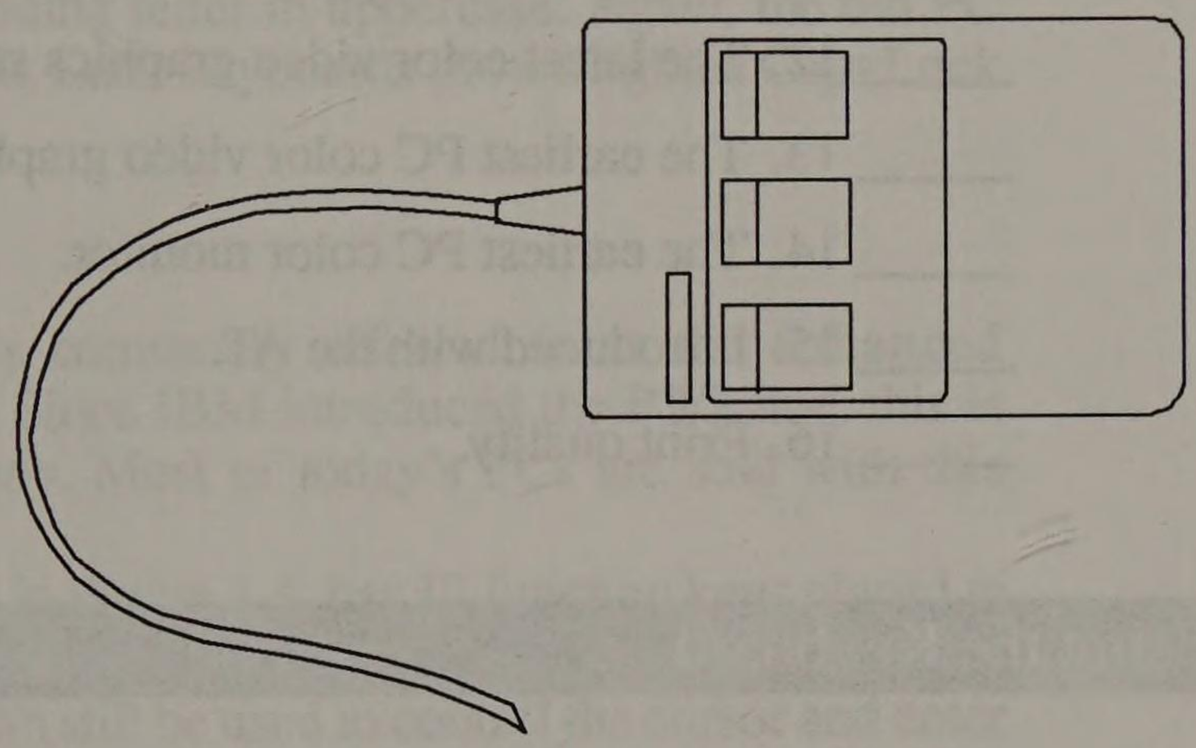


Figure 1.3 A three-button mouse; some mice have two buttons.

A **modem** (MOdulator DEModulator) is a communications device. It can convert (modulate) a computer's digital (0 or 1) signals to analog (sound wave) signals, which can be transmitted via the telephone line, and convert (demodulate) them back to digital signals that a receiving computer can understand. Like a mouse, a modem can be connected to an expansion slot (internal modem) or a serial port (external modem). With a proper communications software package, a modem can be used to communicate with a computer bulletin board available in many large cities, a database service provider such as CompuServe, and other properly equipped PCs.

Modems have undergone major changes. Prices are falling while speed is increasing. Only a few years ago, the 300-**baud** (bits per second) modem was standard. It was quickly replaced by the 1200-baud model, which is rapidly being rendered obsolete by the 2400-baud model. The next wave will come in the form of 9600-baud, and some will be combined with the capability to fax documents.

Scanners can scan text or graphical images and feed them into a PC. With proper software, the scanned text and images can be recognized and manipulated. Scanners are most often used in desktop publishing.

Drill

Select an item in the list to answer questions 7–11:

a. motherboard, b. system unit, c. peripheral, d. parallel port, e. BIOS.

_____ 7. A physical device connected to a computer.

_____ 8. A metal box containing a PC's vital parts.

_____ 9. A common connector (interface) for a printer.

_____ 10. Controller cards are inserted into its expansion slots.

_____ 11. Provides initial instructions to the PC.

Select an acronym in the list to answer questions 12–16:

a. EGA, b. VGA, c. NLQ, d. CGA, e. RGB.

_____ 12. The latest color video graphics standard.

_____ 13. The earliest PC color video graphics card.

_____ 14. The earliest PC color monitor.

_____ 15. Introduced with the AT.

_____ 16. Print quality.

Using the Keyboard

- **The PC keyboard has gone through two changes. The current standard is called enhanced keyboard and has 101 keys.**
- **Some single key and key combination functions are shown in Tables 1.2 and 1.3.**

A keyboard is an important peripheral device connected to the system unit via a jack (connector) in the back. It is the most commonly used device for giving your PC instructions and data. At the keyboard, you enter a DOS command and give your PC information it can process.

Earlier Keyboards

The PC keyboard has undergone some changes. The first PC came with an 83-key keyboard. The AT has a slightly improved keyboard of 84 keys, with an enlarged Enter key and some lighted panels to show the status of various keyboard conditions. These improvements reduce the chances for making typing errors. The enlarged Enter key is helpful. As you press this key to register a command to DOS, you no longer need to take your eyes off the screen to see where the Enter key is.

The 10 function keys of these earlier models are located in two columns on the left side. The right side of the keyboard features a keypad serving two purposes—cursor control and 10-key entry of numbers.

In the normal state, the keypad is used to control the **cursor** (the little blinking underscore character) in application programs such as spreadsheet or word processor. DOS normally does not recognize most of these keys. Only a couple of them are used for editing a DOS command entered from the keyboard; these will be explained in Chapter 4.

The Num Lock key is a **toggle** key; it can switch between two modes. When it is pressed once, the keypad goes into numeric mode. Pressing a key there displays a corresponding number on the screen. Pressing the Num Lock key another time returns the keypad to cursor-control mode. The original PC keyboard does not tell you what mode this keypad is in. Later keyboards show a lighted Num Lock panel.

The Caps Lock key toggles between uppercase and lowercase for the alphabetic keys; other keys are not affected. When Caps Lock is pressed once, pressing a letter key displays on the screen a corresponding letter in uppercase. Again, the old PC keyboard does not tell you this condition. Later keyboards show a lighted Caps Lock panel.

The Enhanced Keyboard

The current standard PC keyboard is commonly referred to as the **enhanced keyboard.** It has 101 keys altogether. Since IBM introduced the PS/2 line, this is the only keyboard sold by the company. Most of today's PCs are sold with this keyboard.

The enhanced keyboard, as shown in Figure 1.4, has 12 function keys placed at the top row of the keyboard. The numeric keypad (with additional math operator keys) is still located at the far right. It can still be used to control the cursor and enter numbers. An extra Enter key is also added. This can speed up 10-key entry of numbers.

An extra area to the left of the keypad is added for the exclusive purpose of cursor control. Now you can have, if you wish, an exclusive cursor keypad and a separate numeric keypad. This is most convenient when you want to move the cursor and enter numbers, as in using a spreadsheet program.

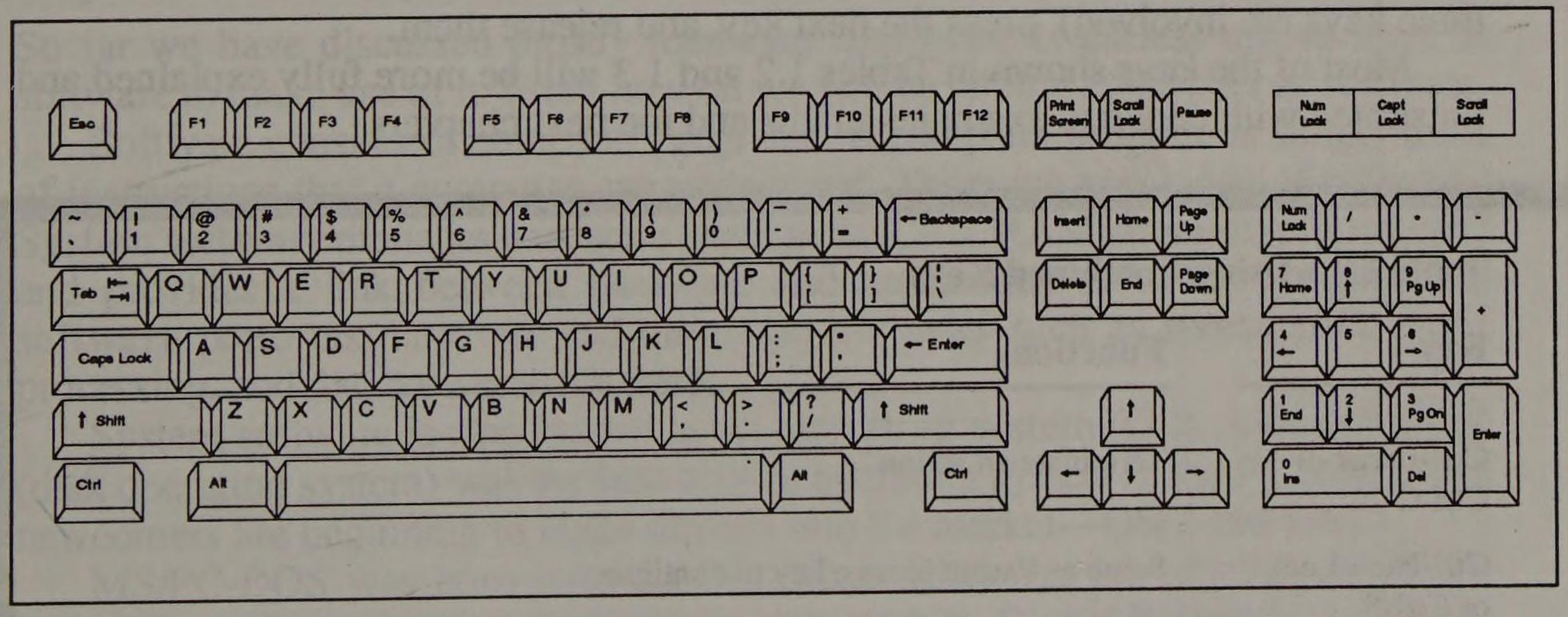


Figure 1.4 The PC (enhanced) keyboard

Using Some Special Keys

Compared to a typewriter, the PC keyboard has many unusual keys. Many application programs take full advantage of them. DOS, however, utilizes very few of these single keys and their numerous combinations. Single keys and their functions are shown in Table 1.2.

Table 1.2 Using Single Keys

Key	Function
Backspace	Erase to the left; same as Left Arrow
Caps Lock	Toggle (switch) between upper- and lowercase
Enter	Register a command
Esc	Abandon a command
Function keys	F1–F6 used for editing; others unused
Num Lock	Toggle keypad between number and cursor
Pause	Halt screen display
Shift	Shift to uppercase for typing a letter

Certain keys can be combined to perform some functions. These key combinations are shown in Table 1.3.

When key combinations are used, hold down the first key (the first two keys if three keys are involved), press the next key, and release them.

Most of the keys shown in Tables 1.2 and 1.3 will be more fully explained and illustrated with concrete examples in this and the next chapters.

Table 1.3 Using Combined Keys

Key	Function
Ctrl-Break or Ctrl-C	Terminate an action
Ctrl-Num Lock or Ctrl-S	Same as Pause; press a key to continue
Ctrl-PrtSc or Ctrl-P	Turn on/off printing; when on, whatever goes to the screen is also printed
Shift-PrtSc	Print a screen display (Shift is not required on a newer keyboard)
Ctrl-Alt-Del	Reboot DOS; same as Reset key if equipped

Application programs (such as WordPerfect and Lotus 1-2-3) utilize many of the keys available on a PC keyboard and their various combinations. You need to learn these specifics when you try to master an application program.

Drill

Select an item in the list to answer questions 17–20:

a. Num Lock, b. Enter, c. Ctrl-C, d. Ctrl-S.

_____ 17. Registers a command.

_____ 18. Halts screen scrolling.

_____ 19. Terminates an action.

_____ 20. Toggles between cursor and numeric modes.

System Software and Application Software

✎ **A computer program to handle hardware is known as an operating system or system software, and one that specializes in a type of task is known as application software. DOS is an operating system.**

So far we have discussed mostly hardware. Hardware is useless unless there is software to make use of it. From now on we will be talking mostly about software.

Software consists of computer programs. A computer program is simply a set of instructions that a computer can understand. There are two kinds of software. **System software** manages hardware (such as the CPU, RAM, monitor, printer, etc.) and provides a link between hardware and application software. **Application software** performs narrower and more specific tasks such as spreadsheet, word processing, and database management.

System software is also known as an **operating system** (OS). Although **DOS** (disk operating system) was the best known operating system in the PC realm, two newcomers are beginning to make inroads into the market—**OS/2** and **UNIX.**

MS/PC-DOS was born with the PC. When IBM was introducing the PC, it looked for an operating system to run its new product. Digital Research, the owner of the then popular operating system called **CP/M** (Control Processor/Micro), was uncooperative. IBM decided to make a deal with an unknown startup company called Microsoft. This connection has transformed a tiny outfit into a multibillion-dollar corporation.

PC-DOS and **MS-DOS** are in most cases identical; both are written by Microsoft. The brand marketed by IBM is called PC-DOS. Other brands licensed to other vendors, such as Compaq, Tandy, Telex, etc., are referred to as MS-DOS. Most of today's PCs, made by IBM or anybody else, can run either brand.

Although there are minor differences between PC and MS brands, there are major changes from an earlier version to a later one.[5] Many people continue to use earlier versions, but the most commonly used versions today are 3.3 and 4. Microsoft continues to market both versions as of 1990.

DOS is basically a single-user, single-tasking operating system. Most of the time only one person can use a computer to do one thing at a time. However, as PCs pervaded the corporate world, the demand for a multiuser, multitasking operating system increased.

Microsoft responded by developing OS/2 (Operating System/2). This operating system allows a properly equipped PC (4MB RAM, 286 or higher CPU) to be used to do several jobs at the same time. Although OS/2 is primarily a multitasking system, with additional hardware and software it can be a multiuser system.

OS/2 competes against UNIX, which was initially developed by AT&T. UNIX has been very popular in academia running minicomputers, which are multiuser systems. Today it is rapidly migrating to the PC world, where it can run multiple connected PCs, each of which can run multiple tasks. Currently, the federal government purchases only UNIX products. This factor may help boost its position in its competition against OS/2.

The availability of application programs is what makes an operating system popular. DOS is so popular today because of practical and well-received application programs such as Lotus 1-2-3 for spreadsheets, WordPerfect for word processing, and dBASE II/III/IV for database management.

Whether OS/2 and UNIX will replace DOS as the standard PC operating system will depend on how many application programs are written for them and how popular these programs become. As of 1990, those for DOS far outnumber those for the other two, and many popular packages are available only for DOS. The position of DOS, despite its many shortcomings, seems secure.

Between application programs and operating systems is another kind of software called **the operating environment.** An operating environment provides an interface between a user and a number of compatible application programs. The user can switch from one program to another or move data from one application to another without exiting any program. The Shell program in WordPerfect's Library package is an example.

One package that has gained increasing popularity is Microsoft Windows. The latest version, 3.0, was released in mid-1990. Used with a 286, 386, or 486 PC, it can provide the graphical "look and feel" of a Macintosh.[6] A user can click a mouse to switch from one program to another or move data back and forth. This should make using a PC much easier for some people.

[5] See Appendix C for the evolution of DOS and how various versions differ from one another.

[6] A lawsuit is pending among Apple, Microsoft, and Xerox as to which company owns the patent to graphical screen interface. Apple sued to prevent Windows' competition against its popular Mac "look and feel." Countersuits were then filed by Xerox and Microsoft.

Starting and Restarting a PC

✎ **Turn on the power switch to start a PC. To restart it, just press the Reset button on the front panel of the system unit. If this is not available, press Ctrl-Alt-Del.**

Depending on the model, each PC is equipped with a red lever located in various parts of the system unit. This lever can be flipped to the upward position marked ON or 1, or to the downward position marked OFF or 0. This lever, like a light switch, turns on or off power to the PC. After power is on, the fan inside the system unit begins to blow to cool off the heat generated by electricity going through the computer. Do not block the outlet for the fan; otherwise, your computer may overheat and malfunction.

If you need to restart the computer because of "boot failure" (or a program causing the computer to freeze), you normally do not need to turn power off and then on, which takes a longer time. Instead, you can just press the red Reset button, usually located on the front panel of the system unit. The fan does not stop blowing. The whole system, however, is started anew. If you press this button while running a program, the program and data residing in RAM will be wiped out.

Tips & Tricks

Surge Protector

If you have your own computer, you may want to invest in a device called a **surge protector.** It may cost as little as $10. It can protect your PC from a sudden power surge caused by lightning and other unexpected occurrences. Without such protection, the fragile electronic parts could overheat, rendering your computer useless.

There are many kinds of surge protectors. The more expensive models usually provide more protection against more violent power surges, but even they are not foolproof. If stormy weather is coming, the best protection is to unplug the PC's power cord.

A surge protector is usually equipped with multiple electrical outlets, to which you can connect the system unit, printer, monitor, and other peripherals you may have. When the switch on the surge protector is turned on, power is supplied to all the units that are left in the on position. This can save you the trouble of individually turning on each separate unit. When you are through, just press one single switch to shut down the whole system.

Older PCs do not have this Reset button. To reboot your computer without turning off power, you can press this key combination:

Ctrl-Alt-Del

Use two fingers to hold down Ctrl and Alt, and use another finger to press Del. The screen turns blank and the whole system restarts.

Some people refer to turning on power as a cold boot and pressing the Reset (or Ctrl-Alt-Del) button as a warm boot. The former will take more time than the latter, particularly in an older PC. If you have a choice, try a warm boot under most circumstances.

Depending on how your system is set up, you may need to turn on the monitor and printer (if you intend to print something). Some monitors have a knob you can pull out to turn on and push in to turn off. Some are equipped with a knob which you turn to the right to turn on and to the left to turn off. Newer monitors have a little panel, which you press once to turn on and twice to turn off. Most monitor switches are located in the front; others, however, may be hidden in the back or below the screen. If your monitor is not on after a while, search for a knob or panel or switch to turn on (or ask a lab assistant).

Drill

Answer questions 21–25 with either A (for application software) or S (for system software):

_____ 21. UNIX.

_____ 22. WordPerfect.

_____ 23. CP/M.

_____ 24. Lotus 1-2-3.

_____ 25. OS/2.

Booting DOS and Entering Date/Time

✎ **To boot DOS, you need to make a system disk available to your PC. The system files can be stored on a hard disk or a floppy disk.**

✎ **If your PC calendar is backed up with a battery, the date and time data will be kept up to date. If not, you need to supply the correct numbers after DOS is booted.**

When power is supplied to a PC, the ROM BIOS inside the machine starts to work. It may send different diagnostic information to the screen. Then DOS is booted from

a system disk containing hidden system files. If the system files are not available, booting fails and the screen displays an error message. You can insert a bootable disk and press a key to try again.

Hard Disk Users: If you are using a hard disk, it should be set up in such a way that DOS will be booted from the hard disk. In that case, leave the door of drive A open. The light on the floppy drive will turn on for a while. Then the BIOS will go to drive C to try to boot from there.

If you have no hard disk or just want to boot from a floppy disk, insert a boot disk in drive A and turn on the power. A boot disk requires certain special files. You can create a boot disk by using the `FORMAT` command. This will be explained in Chapter 3.

Tips & Tricks

Disk Drive Warning Red Light

When DOS reads data from or writes data to a disk, the red light on the front panel lights up and you can hear the spinning noise coming from the drive. Do not disturb the drive by opening the door or taking out the disk. If you do, you could cause reading or writing failure and possible damage to your disk. If you want to take out a disk and exchange it, do so after the light goes off and the spinning stops.

On some occasions, you may encounter an out-of-control situation of perpetual disk spinning. You can try to stop it by pressing Ctrl-C (or Ctrl-Break; hold down Ctrl and press either key). If this does not work, you may need to take out the disk and reboot DOS.

If DOS succeeds in booting, a two-line message appears:

```
Current date is Wed 06-20-1990
Enter new date (mm-dd-yy): _
```

The current date read by DOS is displayed in the first line. In the second line, you are prompted to enter a correct date after the colon.

If your computer is equipped with a battery-powered calendar, a correct calendar will be maintained as long as the battery supplies the power to keep the calendar up to date. In that case, just press Enter to go to the next message. If your computer is not so equipped, the current date read by DOS may be this:

```
Current date is Tue 01-01-1980
Enter new date (mm-dd-yy): _
```

If this is the case, you need to enter the correct two-digit month, date, and year, such as:

```
Enter new date (mm-dd-yy): 06-20-90
```

Each item has two numeric digits and is separated from another item with a hyphen. Do not worry about the day of the week; DOS will take care of it.

After you enter a correct date format or just press Enter without typing a date, another two-line message appears:

```
Current time is  7:49:30.68a
Enter new time: _
```

Again, if the time is correct, just press Enter. If not, you need to enter hour, colon, minute, colon, second and optional fractions. (See the next section for more details.)

Tips & Tricks

Date and Time

It is not vital to keep an accurate date and time in your PC. It will function even with wrong information.

It is, however, a good practice to keep your PC up to date (and time). Many benefits can be derived from accurate date-time data. When you save a file, DOS saves the file's creation date and time. You can use some DOS commands to show a file's creation date-time and others to copy files marked with a specific date.

After both date and time are taken care of, a message like the following will appear:

```
Microsoft (R) MS-DOS(R) Version 4.01
(C)Copyright Microsoft Corp 1981-1988
A>_
```

It shows what DOS brand (PC or MS) and version number you are using. The DOS prompt (A>) appears with the cursor after it. This cursor moves as you type or do something else. DOS is now booted and ready to accept your command.

The DOS Prompt and Command Line

- **As soon as DOS is booted, a drive letter followed by a > (greater than) sign appears. This is the DOS prompt.**
- **The line where the DOS prompt appears is the command line for entering a DOS command.**

✎ **Use the `CLS` command to clear the screen, `VER` to show the DOS version, `DATE` and `TIME` to show and change date and time information.**

Depending on where DOS is booted from, you will see different displays after booting. If you boot from a floppy drive, this appears:

```
A>_
```

If you boot from a hard disk, you see this instead:

```
C>
```

This is the **DOS prompt,** or **system prompt.** It consists of two elements: the drive letter and the > (greater than) sign. You are now prompted to enter a **DOS command,** which is an instruction DOS can understand and implement.

Tips & Tricks

Correcting Typing Errors

If you make a mistake while typing something on the command line, use the Backspace or Left Arrow key to move the cursor left and at the same time erase a typed character. After erasing an error, retype your command.

If you want to abandon a line before pressing Enter, just press the Esc key. The cursor will move down one line for you to enter a new command.

DOS uses a number of function keys as editing keys. Chapter 4 explains how to use them.

The line where the DOS prompt appears is referred to as the **command line.** Here you are to enter a command by typing something DOS understands and pressing Enter to register it. The little blinking underscore character is referred to as the cursor. It marks the position where a command is to be entered. The cursor position changes as commands are typed and entered.

Now you are going to enter some commands to instruct DOS to do some simple chores. Follow these steps:

1. Just press Enter:

```
A>
A>_
```

If you just press Enter without typing anything, the cursor moves down one line and the DOS prompt is displayed again. Nothing else happens.

Hard Disk Users: If you use a hard disk, treat A> as C>. From now on, we will use A> as the default prompt where you are expected to enter a command.

2. Clear the screen:

```
A>cls
A>_
```

The screen should now be cleared of everything except the DOS prompt located at the top left corner of the screen. Any time you want to clear a cluttered screen and locate the cursor at the top left corner of the screen, just enter the CLS command.

Tips & Tricks

Upper- or Lowercase

Should you enter a DOS command in uppercase or lowercase? It doesn't matter. DOS automatically converts everything to uppercase. Thus the three commands below are treated as identical:

```
A>cls
A>Cls
A>CLS
```

There are some rare occasions when uppercase or lowercase is expected. On these occasions, you will be reminded to enter the correct case.

3. Find out the DOS version:

```
A>ver
MS-DOS Version 4.01
A>_
```

By entering the VER command, you ask DOS to tell you what brand and version you are using. (Your display may be different from the above.) The first number signifies a major revision, the second a minor revision, and the third a special release (such as 2.11 being a 2.1 designed specifically for the PC Junior) or a bug-fix release. Version 4.01, the latest as of this writing, is a release primarily to fix the bugs (errors) in 4.0.

4. Display the current date:

```
A>date
Current date is Wed 06-20-90
Enter new date (mm-dd-yy): _
```

5. Enter month-date:

```
Enter new date (mm-dd-yy): 06-21
Invalid date
Enter new date (mm-dd-yy): _
```

You must enter two-digit numbers for all three items of month (mm), date (dd), and year (yy); hyphens must separate them. If you miss one, as shown above, DOS displays the "Invalid date" message and prompts you to enter another.

6. Enter month-date-year:

```
Enter new date (mm-dd-yy): 06-21-90
A>_
```

The new date has been accepted and the system prompt reappears. If you neglected to enter a correct date right after booting, you can use the above steps to change the current date.

7. Display the current time and enter a new time:

```
A>time
Current time is  3:37:55.39p
Enter new time: 3:40p
A>_
```

The above entry contains only hour and minute, plus the "p" for P.M. The second numbers can be omitted if you wish.

Version Variances: The "p" (for P.M.) or "a" (for A.M.) notation is not permitted prior to version 4. In that case, you must enter time in the 24-hour format.

8. Enter the current time in 24-hour format:

```
A>time
Current time is  3:42:02.71p
Enter new time: 15:43
A>_
```

9. Display the current time without changing it:

```
A>time
Current time is  3:45:02.54p
Enter new time: _
A>
```

If you just press Enter when prompted to enter a new time, the old time is kept. If you use version 3.3 or earlier, the above display will appear as 15:45:02.54.

Tips & Tricks

The PC Calendar

Each PC has an internal built-in calendar. When power is on, it functions like a digital watch to keep track of date and time data. If your PC has no battery to back up this calendar, it will go back to 01-01-80 when power is turned off. If your PC has a battery-powered calendar, it will keep track of the date and time even when power is off as long as the battery is in good condition.

If you have a newer PC, using DATE and TIME commands will affect the calendar. The new numbers are entered into the system and kept up to date as long as the battery is working.

If you have an old IBM PC, the DATE and TIME commands do not affect the system even if the calendar is powered by a battery. To change the system data, you need to use the software bundled with the hardware like the AST Sixpack multifunction card.

Practice 1-1

- ☐ 1. Reboot DOS without turning off power.
- ☐ 2. On the command line, type something and erase it.
- ☐ 3. On the command line, type something and abandon it.
- ☐ 4. Clear the screen.
- ☐ 5. Show the DOS version.
- ☐ 6. Show the current date without changing it.
- ☐ 7. Change the current time by one minute.

Ending a DOS Session

✎ **When you are through working with your PC, turn off power supply to the system unit and turn off peripherals such as monitor and printer.**

When you are done with your PC, you need to turn off its power supply. The power supply to each connected device must also be shut off. Otherwise, it may burn out

or prematurely wear out. If the PC is in a lab used by others, you may leave it on for other people. (Ask a lab assistant about the lab's policy and your responsibilities.)

If you have all the peripherals connected to a multiple-outlet device, turning off the switch will shut off power to every connected device. This is the simplest and most efficient arrangement.

Warning: Your PC may share the same power supply with another PC. In that case, don't switch off the power if the other PC is being used.

Do not forget to turn off the monitor if its power supply is not shut off. Leaving a monitor on for a long period of time will result in characters permanently etched to the screen. Such a monitor may no longer be usable or salvageable.

Do not cut off power supply in the middle of running a program. Everything in RAM will be lost. Most application programs, such as WordPerfect, prescribe a procedure for an orderly exit. It is a good idea to follow such a procedure.

Generally speaking, when the DOS prompt appears, you are not running any program and can thus safely shut off power.

If you boot from a floppy disk, remember to remove it before or after shutting off power. After power is shut off (if you are responsible for turning on and shutting off), you can safely walk away with your own disks.

Tips & Tricks

Computer Radiation

The widespread use of computers has stirred up controversy over how much radiation is being emitted from a computer and how harmful such radiation is. Many office workers who work with computers all day have complained about cancer, miscarriages, and other harms allegedly caused by computer radiation.

The PC can emit harmful radiation from the monitor and system unit. You should take these precautions to protect yourself against potential hazards:

- **When working with a PC, keep yourself as far away from the monitor and system as possible. The keyboard cord can be stretched for this purpose.**
- **Turn off the PC whenever it is not being used for a sustained period of time.**

Drill

True or false?

_____ 26. If your PC has no hard disk, you need to insert a system (boot) disk in drive A to start DOS.

_____ 27. A PC can maintain a correct calendar only when it is equipped with a battery.

_____ 28. A DOS command must be entered in uppercase.

_____ 29. The `CLS` command clears the entire screen, including the DOS prompt.

_____ 30. The `DATE` and `TIME` commands can be used to display as well as change a PC's date and time.

Summary and Review

Check each item as you review and understand the following highlights of the chapter.

☐ 1. Depending on sizes and capabilities, computers can be grouped into mainframes, minis, and PCs. PCs include XT, AT, PS/1, and PS/2. Many PS/2 models use IBM's new MCA bus and some new ATs are equipped with the new EISA bus.

☐ 2. A PC's capability is determined mostly by its CPU. Most CPUs used in today's PCs originated from Intel Corporation. These include 8088/8086, 80286, 80386, and 80486. The higher a CPU's number, the more powerful it is.

☐ 3. A typical PC system consists of a system unit, a keyboard, a monitor, and a printer. The system unit contains one or more disk drives and a motherboard to which a CPU, ROM and RAM chips, and peripheral devices are attached.

☐ 4. There are various types of monitors available today, each of which can be driven by a matching graphics card: Hercules for a monochrome monitor, CGA for an RGB monitor, EGA for an EGA monitor, and VGA for a VGA monitor.

☐ 5. Three common types of printers are in use today: daisy wheel, dot matrix, and laser. Daisy wheels have mostly been replaced by dot matrixes. With excellent print quality and falling prices, lasers are increasing in popularity.

☐ 6. There are two major types of PC keyboards. The first was introduced with the original IBM PC and was slightly improved when the AT was introduced. There are 10 function keys located at the left end. The keypad on the right

side can be toggled between cursor and numeric modes. The enhanced keyboard that was introduced with the PS/2 has 12 function keys located at the top and an additional cursor pad in the middle.

☐ 7. There are two major categories of computer software: system and application. The former is also known as the operating system, or OS for short. DOS is an OS. There are two brands of DOS—PC-DOS and MS-DOS, both written by Microsoft Corporation. The former is licensed to IBM and the latter to other PC vendors. The two are practically identical.

☐ 8. Three major operating systems are currently competing in the PC market: DOS, OS/2, and UNIX. DOS practically monopolizes the single-user and single-tasking market. The new OS/2 (also from Microsoft) and UNIX, which is transferred from the mini world, compete in the multiuser and multitasking environment.

☐ 9. When you start your PC, you need to have DOS system files available before you switch on power. They could be stored on the hard disk. In that case, leave the door of drive A open when power is switched on. If you have no hard disk, you need to insert a system disk in drive A and close its door before switching on power.

☐ 10. If your PC is locked up (frozen) and will no longer respond to a command from the keyboard, you need to reboot the system. Normally you do not need to switch power off and then on again. This takes a long time to restart. Instead, you can just press the Reset button. If your PC is not equipped with it, press the key combination Ctrl-Alt-Del.

☐ 11. After DOS is booted, you will be asked to enter the date and time. If they are correct, you can just press Enter to keep the old information. If your PC has a battery-backed calendar, the information will be kept up to date. If not, you need to enter the current numbers.

☐ 12. The `DATE` and `TIME` commands can be used to display as well as change the date and time information.

☐ 13. Use the `CLS` and `VER` commands to respectively clear the screen and display the DOS version.

☐ 14. When you have finished using your PC, you need to shut down the whole system. Power supply to the system unit must be switched off. Other independently powered devices also need to be turned off.

Review Questions

Write an answer for each question below.

1. Briefly define these terms: PC, XT, AT, PS/1, and PS/2.
2. Explain and relate these terms: bus, MCA, and EISA.

3. Find out what CPU is used in your PC. How does it compare to the others?
4. What major components are located inside a system unit? What do they do?
5. What kind of monitor do you use? How does it compare to the other available models?
6. What kind of printer are you using? Describe its capabilities.
7. What kind of keyboard are you using? Describe its major sections.
8. What does a modem do?
9. What does the Ctrl-Alt-Del key combination do?
10. What is the difference between an operating system and an application program?
11. What are the three major operating systems for the PC today? How do they differ?
12. What is a surge protector?
13. Does your PC have a battery-backed calendar? How do you know? What purpose does it serve?
14. What DOS brand and version do you use? How do you find out?
15. If you neglect to enter the correct date and time right after DOS is booted, how can you change them later?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
`TIME`	When entering a time, you must use the 24-hour format for P.M. times.	You can use "a" for A.M. and "p" for P.M., or the 24-hour format.

A Quick Tour Through DOS

Chapter 2

TOPICS:

What Is a File?
Changing the Default Drive
Displaying Directory Entries
Halting a Screen Display
Displaying a File's Contents
Copying Files
Renaming Files
Deleting Files
Copying a Disk
Internal and External Commands
Printing a Screen Display and Disk File

COMMANDS COVERED:

`DIR, TYPE, COPY, REN, DEL (ERASE), DISKCOPY`

NEW TERMS AND SYMBOLS:

Files, ASCII files, executable files, batch files, default drive, directory entries, switch, parameter, source, target, syntax, path, path name, internal/external command, printout

REQUIRED MATERIALS:

1. Two practice disks (at least one formatted)
2. The disk accompanying this book (DOS-TA)
3. Access to `DISKCOPY.COM`

Quick Start

✎ **To change from one drive to another:**

```
A>b:
```

✎ **To show directory information:**

```
A>dir
A>dir filename
```

✎ **To halt (pause) screen display:**
Press Pause (if any), Ctrl-S, or Ctrl-Num Lock.

✎ **To show a file's contents:**

```
A>type filename
```

✎ **To copy a file to another disk without a name change:**

```
A>copy filename b:
```

✎ **To copy a file to another disk with a new name:**

```
A>copy oldname b:newname
```

✎ **To copy a file to the same disk with a new name:**

```
A>copy oldname newname
```

✎ **To rename a file:**

```
A>ren oldname newname
```

✎ **To delete a file:**

```
A>del filename
```

✎ **To copy a disk:**

```
A>diskcopy a: b:
```

✎ **To print a screen display:**
Press Shift-Print Scrn.

✎ **To display and print simultaneously:**
Press Ctrl-P (or Ctrl-Print Scrn).

✎ **To print a disk file:**

```
A>copy filename prn
```

✎ **To redirect output from the screen to the printer:**

```
A>dir > prn
A>type filename > prn
```

Consider this chapter a DOS survival course. It will take you, step by step, through some of the most commonly used DOS commands. These commands are rather easy to understand and use. After you become familiar with these commands, you should be on your way to success with DOS.

Some of these commands have more exotic details. They will be covered in later chapters. At this time, you need to concentrate on the most common applications.

You should have with you the disk labeled DOS-TA that comes with this book. The write-protect notch of the disk (a little square opening at the upper right corner) should be covered with a tab. This protects the disk and prevents you from inadvertently erasing the existing files.

You should also have two floppy disks ready for use. One must be formatted, with some disk space left for storing new files. The other can be new and unformatted, or formatted and used; in the latter case, it must not contain any file you intend to save.

Tips & Tricks

Quick Formatting

If you need to format a floppy disk at this time, follow these steps:

1. Use `DIR` to find out where the `FORMAT.COM` file is. If your PC has a hard disk, it is likely to be stored in the DOS directory. If you have a floppy system, swap disks as necessary; keep in drive A the disk containing the file.

2. Insert in drive B a disk to be formatted. (Put it in drive A instead if you have a hard disk and only one floppy drive.)

3. Enter an appropriate command below:

```
A>format b:                 (floppy system)
A>c:\dos\format a:          (hard disk, format drive A)
A>c:\dos\format b:          (hard disk, format drive B)
```

4. Press Enter when instructed to insert a diskette. Wait for the running message to stop. Press Enter if instructed to enter a volume label. When asked "Format another (Y/N)?", type N and press Enter. (If you want to format another, enter Y instead.)

Chapter 3 provides much more detail.

What Is a File?

- **Use file name extensions to distinguish some common types of files.**
- **DOS executable files have three common file name extensions: COM, EXE, and BAT.**

Many DOS commands, including those discussed in this chapter, are used to maneuver files. Before you can do that, you need to be familiar with a few related terms.

A **file** is a collection of information (data), that a computer can recognize and maneuver. It can be loaded to memory or stored on disk with a unique name.

DOS disks contain dozens of files. Each file has a file name, plus an optional extension of up to three characters placed after a period. The file name tells you the purpose of a file; the extension describes the type of file.

ASCII Files

A file may contain nothing but ASCII characters (see Appendixes B and F). Such a file is known as a text file, DOS text file, or **ASCII file**. You can use the `TYPE` command, as explained below, to clearly show the contents of such a file. Users commonly use file name extensions such as `DOC` (document) or `TXT` (text) to signify an ASCII file, although an extension is not mandatory. You will learn to create such files and use proper file names and extensions in Chapter 4.

Version Variances: Unlike its predecessors, version 4 contains quite a few text files carrying the extension TXT. These provide additional information (not available in the printed manual) for certain commands. `GRAPHICS.TXT` is an example. If you use version 4, look for a file named `README.TXT` on the Operating 3 disk to see the latest additions and corrections.

Tips & Tricks

`README` Files

When you see a disk file named `README`, `READ.ME`, `README.DOC`, or `README.TXT`, you should use the TYPE command to display its contents on the screen.

Software publishers usually prepare printed manuals long before releasing a software package. Manuals take a long time to prepare and print. After manuals are printed, programmers may add new features. Since manuals cannot be altered easily, vendors simply include a README file on disk. When you buy a package, you should look for such a file to learn last-minute additions or instructions.

A file created with a word processor usually contains formatting codes, which control text characteristics such as boldface, center, and flush right. Such a file is not a pure ASCII file. If you use TYPE to display its contents, you will find some recognizable characters and many strange-looking symbols.

COM and EXE Files

A file can contain program code that a computer could execute. Such a file is known as an **executable file** or binary file. It is created with a programming language. It can have the file name extension COM or EXE, which is usually determined by the language you use to create such a program. Many external DOS commands, as explained below, are separate programs. Some programs have the extension COM and others have EXE. When you find files with these extensions on your DOS disks, you can execute them by entering their file names with or without an extension.

Batch Files

Another kind of executable file is called a **batch file.** It is an ASCII file containing commands DOS can understand and execute. Batch files carry the extension BAT. Batch files are important because they can save you a lot of time. You will learn more about them starting in Chapter 4.

SYS Files

SYS files, such as DISPLAY.SYS and VDISK.SYS (RAMDRIVE.SYS for MS-DOS) are device drivers. When installed, they enable DOS to handle uncommon devices. Chapter 9 tells you how to install these drivers.

Table 2.1 sums up the ways to distinguish the numerous DOS files by their extensions.

Table 2.1 Common DOS File Name Extensions

Extensions	Purpose
BAT	Batch file, executable ASCII text
COM	Command (executable) file, binary
EXE	Executable file, binary
SYS	System (device) driver file, binary
TXT	Text (ASCII) file

Changing the Default Drive

- **The DOS prompt automatically shows the letter of the default drive.**
- **Change the default drive to a new one by entering a drive letter followed by a colon.**

When DOS is booted, the system prompt shows the letter of the default drive and the > (greater than) sign. The **default drive** (also known as current drive) is where a DOS command will look for a file—unless preceded or followed by a drive letter (as explained in the next section).

Note: Before you perform the following steps, you must boot DOS as explained in Chapter 1.

The default drive can be changed by entering a letter followed by a colon. Follow these steps to change default drives:

1. Change from A to B, or make B the default drive:[1]

```
A>b:
B>_
```

You must end with a colon. If not, the default drive will not be changed and you will get this error message instead:

```
A>b
Bad command or file name
A>_
```

Here DOS interprets your move as instructing it to run a program named B (B.COM, B.EXE, or B.BAT). DOS will go to drive A (the default drive) to search for it. If such a program exists, it will be executed. If not, the above message will appear and the default drive will not change.

2. Change from B to A, or make A the default drive:

```
B>a:
A>_
```

3. Change to C:

```
A>c:
C>_
```

If you have a hard disk, the DOS prompt shows the new drive's letter. If not, you will get this error message and no change of the original drive letter:

```
A>c:
Invalid drive specification
A>_
```

[1] If your PC has only one floppy drive installed, DOS will recognize it as both A and B. You can use the drive as A or B.

4. Return to A (if you have a hard disk):

```
C>a:
A>_
```

Drill

Each file in questions 1–6 is executable. True or false?

_____ 1. `README.EXE`

_____ 2. `README.TXT`

_____ 3. `README.BAT`

_____ 4. `ANSI.SYS`

_____ 5. `COMMAND.DOC`

_____ 6. `COMMAND.COM`

_____ 7. DOS interprets the command `A>c` as meaning that you want to:
   a. go to drive C
   b. run an executable file
   c. cancel an action
   d. copy a file

_____ 8. If a file named `B.COM` exists on the disk in drive A, the command `A>b:` will:
   a. make B the default drive
   b. execute `B.COM`
   c. display an error message
   d. do nothing

Displaying Directory Entries

- **Use `DIR` alone to display the default drive's directory entries.**
- **Use `DIR` with a drive letter (parameter) to display another drive's directory entries.**
- **Use `DIR` with a specific file name to display the file's directory information.**
- **Use `DIR` with the /P or /W switch if there are many files.**

The `DIR` command is used to display on the screen the **directory entries** of a disk or directory. These entries include subdirectories, if any, and file names. Directories

and subdirectories will be explained in Chapter 5. Here we only demonstrate the use of DIR. Follow these steps:

1. Insert the DOS-TA disk in drive A.

2. Show the directory entries of the disk in the default drive:

```
A>dir
 Volume in drive A is DOS-TA ———————————— Volume label
 Volume Serial Number is 3D65-18CE ————————— Serial number
 Directory of  A:\ ———————————————————————— Directory

DOS      EXE    49119 06-24-90     10:51p ——— File time
DRILL    EXE    30192 06-14-90     8:10p
MENU     EXE    12861 06-13-90     5:06p
MENUB    BAT     2902 06-25-90     3:13p
DOSLES1          3471 06-03-90     7:44p
DOSLES2          5582 06-04-90     9:45p
. . .
KEY      COM        8 06-22-89     5:31p
SAMPLE   TXT      268 06-25-90     3:30p
        18 File(s)184320 bytes free ———————— Disk space
                                             Number of files
```

File name
Extension
File size
File date

There are three distinct parts in this display. At the top is disk information. The first line shows the volume label for this disk; you will learn to create and change volume labels in Chapter 3. The second line shows the serial number, which is available only for a disk formatted with version 4; this will also be explained in Chapter 3. The third line shows the default drive's letter and the current directory, which will be covered in Chapter 5.

The middle part displays files in five columns: (1) file name, (2) file name extension, (3) file size in bytes, (4) file creation date, and (5) file creation time.

The last part, containing a single line, shows the total number of files and the amount of available disk space.

3. Go to drive B:

```
A>b:
B>_
```

4. Show the directory entries of the disk in drive A:[2]

```
B>dir a:
```

[2] If you have only one floppy drive, this message appears:

```
Insert diskette for drive A: and press any key when ready
```

If you want to show another disk, put in a new one and press a key to continue. Otherwise, just press a key.

The drive letter after the `DIR` command tells DOS to act on that particular drive rather than the default drive. If this drive parameter is omitted, DOS will act on the default drive, as shown in step 2. From now on, when you want a DOS command to act on a drive other than the default drive, just specify a drive letter after the command.

Tips & Tricks

What Is a Parameter?

Some DOS commands require one or two **parameters** to complete a job. A parameter is a qualifier or argument (command line argument) that defines the way a command is to do its job.

You must provide parameters as expected. If a wrong parameter is used, you will get the "Invalid parameter" message. If more parameters than expected are entered, you will get another error message, like this:

```
A>dir a: b:
Too many parameters - b:
```

In some cases, a parameter can be omitted. DOS then uses certain default values. Consider this example:

```
A>dir a:
```

The parameter `a:` can be omitted. If so, the default drive is acted on. If you want the `DIR` command to act on another drive, you will have to specify a parameter, such as:

```
A>dir b:
```

A parameter must not be separated by a space; otherwise DOS will treat it as two parameters. Consider this:

```
A>dir b: filename
Too many parameters - filename
```

Without a space, `B:FILENAME` is treated as a single parameter. The extra space makes it two parameters. Since `DIR` expects one parameter, the error message results.

5. Show the directory information of a file in another drive:

```
B>dir a:sample.txt
 Volume in drive A is DOS-TA
 Volume Serial Number is 3D65-18CE
 Directory of  A:\
SAMPLE    TXT 268    06-25-90     3:30p
        1 File(s)   184320 bytes free
```

The top and bottom portions are the same, but the middle portion shows only one file.

6. Go back to drive A:

```
B>a:
A>_
```

7. Show the directory entries in the default drive in the wide format by specifying the /W switch:

```
A>dir /w
Volume in drive A is DOS-TA
Volume Serial Number is 3D65-18CE
Directory of  A:\
DOS        EXE     DRILL      EXE     MENU       EXE     MENUB      BAT     DOSLES1
DOSLES2            DOSLES10           DOSLES11           DOSLES12           DOSLES3
DOSLES4            DOSLES5            DOSLES6            DOSLES7            DOSLES8
DOSLES9            KEY        COM     SAMPLE     TXT
      18 File(s)      184320 bytes free
```

The top and bottom portions remain the same as when you use `DIR` alone. The middle portion, however, shows only file names and extensions in five columns. File size, date, and time data are omitted.

8. Specify the /P switch:

```
A>dir /p
```

This tells DOS to display the directory entries of drive A one screen (page) at a time and, if there is more to show, pause for you to press a key to continue.

`DIR` has two switches, /P (pause or page) and /W (wide). They are useful if you have a lot of files to display. If there is a long list of files, using the /P switch displays this message:

```
Press any key to continue . . .
```

When you press a key, another screen appears, and so on until the end of the display. If you press Ctrl-C (or Ctrl-Break; hold down Ctrl and press either C or Break) after this message appears, the screen shows `^C` and further display of directory entries is terminated. So, if you have found what you are looking for and want no more display, just press Ctrl-C so as not to waste any more time.

Tips & Tricks

What Is a Switch?

A **switch** is an option that can be used with a specific DOS command. Some commands have no switch, while others have more than half a dozen.

A switch enables a command to do something extra or behave in a slightly different manner. In order to make full use of a complex command, you need to know all its available switches.

A switch is placed after a command and immediately preceded by a slash. A space before the slash is advisable. The lack of a space may cause a problem under some circumstances. A space after the slash is not permitted. Consider the following:

```
A>dir / p
Invalid switch - /
```

The error message is caused by the space between the slash and the switch letter. The space makes DOS interpret your command as using a switch consisting of a single slash, which is not a valid switch.

A switch can be placed before or after a parameter. Both arrangements below are legal:

```
A>dir /w b:
A>dir b: /w
```

Practice 2-1

- ☐ 1. Go from drive A to B and back to drive A.
- ☐ 2. Display the directory entries of the disk in the default drive.
- ☐ 3. With A as the default drive, display in the wide format the directory entries of the disk in drive B.
- ☐ 4. With B as the default drive, display the directory information of a single file in drive A.

Tips & Tricks

Drive Failure

If DOS cannot access a disk, you will be given a number of options. Suppose you leave the drive B door open (or insert no disk) and enter this command:

```
A>dir b:
Not ready reading drive B
Abort, Retry, Fail?_
```

If you want to abort the command, just press A; the DOS prompt reappears. If you want to retry, fix the drive and press R. If you press F, DOS displays a message showing the cause of the failure.

Versions before 3.3 show `Ignore` instead of `Fail`. Under some circumstances, version 4 also shows the `Ignore` option, like this:

```
Abort, Retry, Ignore, Fail?_
```

The `Ignore` option tells DOS to ignore the encountered error and move forward.

Drill

Each item in questions 9–14 is displayed when `DIR` is used without a switch or parameter. True or false?

_____ 9. The total number of files.

_____ 10. The amount of available memory.

_____ 11. The letter for the drive.

_____ 12. The date and time of each file.

_____ 13. The size of each file.

_____ 14. Available disk space.

_____ 15. When the /W switch is used with `DIR`, the screen shows:
   a. each file's date and time
   b. 5 columns of file names with extensions
   c. the total number of files
   d. both b and c

_____ 16. In the command `A>dir b:filename`, "`b:filename`" is known as:
a. a switch
b. an extension
c. a parameter
d. a tail

Halting a Screen Display

✎ **Press the Pause key to halt a screen display. If your keyboard lacks this key, use Ctrl-S or Ctrl-Num Lock. Press the space bar or another key to continue scrolling (showing more).**

When you use a command to display something that is longer than one screen length, the display quickly scrolls out of sight. To stop the scrolling so that you can examine it, try one of the following: the Pause key, the Ctrl-S combination, or the Ctrl-Num Lock combination.

The 101-key enhanced keyboard has a special key named Pause that halts the screen display. Follow these steps:

1. Insert the DOS-TA disk in drive A (if it is not already there).
2. Display the directory entries:

   ```
   A>dir
   ```

3. Press Pause.

If you press the Pause key before everything is displayed, the scrolling is suspended.

4. Press the space bar (or any other key) to continue.

If you press Ctrl-C (or Ctrl-Break), the display is terminated and the DOS prompt reappears.

If you have an older keyboard without the Pause key, you can use Ctrl-S or Ctrl-Num Lock instead (hold down Ctrl and press either key).

The Pause key is especially useful when you use the `TYPE` command to display a long text file, as explained below.

Displaying a File's Contents

- ✎ **Use `TYPE` followed by a file name to display a file's contents on the screen.**
- ✎ **Use a path name when you want `TYPE` or another DOS command to act on a file located in another drive.**

The `TYPE` command is used to display on the screen a text file stored on a disk. Follow these steps to use `TYPE`:

1. Insert the DOS-TA disk in drive A (if it is not already there).
2. Show the contents of a file in the default drive:

```
A>type sample.txt
This is a sample text file. It is to be used for
demonstrating some DOS commands.
This file has no importance. It can be copied,
altered, and deleted after you have made use
of it.
You will learn in later chapters how to create
and alter a file like this.
```

The entire contents of the text file are now displayed on the screen.

3. Halt scrolling by pressing the Pause key (or Ctrl-S) after entering this command:

```
A>type sample.txt
```

Depending on how soon you press the Pause key, the writing of text lines to the screen is halted. When you press a key again, scrolling and writing to the screen continues. This process continues to the end of the file.

4. Go to drive B:

```
A>b:
B>_
```

5. Show the contents of a file in another drive:

```
B>type a:sample.txt
```

This command tells DOS to use the `TYPE` command to display the contents of the specified file in drive A. The screen should display the same text lines shown after step 2.

Notice that every time you use `TYPE`, you have to specify a single parameter. This parameter must be a file—unlike `DIR`, for which you can specify a drive letter or a file. If you specify a drive letter alone, you will get an error message. Consider the following:

```
B>type a:
File not found - A:
```

DOS thinks you want to display a file named a: . It will simply reject your command without even bothering to access a disk.

Tips & Tricks

Paths and Path Names

When you want a DOS command to act on a file located in the default drive, you need to specify only the name of the file after the command, such as:

```
A>type sample.txt
```

If you want a command to act on a file located somewhere else, you need to precede the file name with a **path** (drive/directory) prefix so that DOS knows where to find the file, such as:

```
A>type b:sample.txt
```

In this case, DOS will go to the specified B drive to act on the specified file. This combination is commonly referred to as a **path name**—combining path and file name. Another example of path name (or full path name) was shown at the beginning of this chapter:

```
A>c:\dos\format a:
```

Copying Files

- **Use `COPY` to copy individual files from one disk to another, or make a duplicate (with a different name) on the same disk.**
- **`COPY` expects two parameters, the source and the target.**
- **A target file name can be the same as or different from the name of the source file.**
- **When only one parameter is specified, it is presumed to be the source; the default drive is assumed to be the target.**

Copying files is a vital part of using a computer. Although Chapter 6 will provide much more detail, this section gives you some simple examples so that you can start using this function as soon as possible.

Follow these steps to create copies of a file:

1. Insert the DOS-TA disk in drive A and a practice (formatted) disk in drive B.[3]
2. Copy a file from A to B:[4]

```
A>copy sample.txt b:
        1 File(s) copied
```

Notice that the message tells you that one file has been copied. If there is a file in drive B with the exact name, its contents and date-time data will be replaced with the copy coming from drive A. This will be done without any warning.

3. Show the new directory entry:

```
A>dir b:sample.txt
```

You should see, among other things, this item:

```
SAMPLE  TXT    268    06-25-90     3:30p
```

This proves that a new file was created after you used the COPY command.

Now peel off the write-protect tab from the DOS-TA disk if one is there. We'll put it back later. Now we are going to write to this disk.

4. Copy a file without a target:

```
A>copy b:sample.txt
```

This command tells DOS to use COPY to copy the specified file from drive B to the omitted default drive, namely A. This command is the same as the one below:

```
A>copy b:sample.txt a:
```

The original copy in drive A matching the specified name is now replaced with the one from drive B.

5. Go to drive B:

```
A>b:

B>_
```

6. Copy from B to A:

```
B>copy sample.txt a:
```

This does the same thing as step 4, namely, copying a file from drive B to drive A. In this example, the target parameter of a: is required. Without it, an error message will appear, as shown below in step 7.

7. Create a duplicate file on the same disk:

```
B>copy sample.txt sample.txt
File cannot be copied onto itself
        0 File(s) copied
```

[3] If you have a single floppy drive, put the DOS-TA disk in it first and follow the steps below. DOS will instruct you to swap disks.

[4] If you have only one floppy drive, replace the DOS-TA disk with your own after the message below appears; swap disks in the future in similar situations:

```
Insert diskette for drive B: and press any key when ready
```

Tips & Tricks

Source and Target

Some commands, including `COPY`, expect two parameters, source and target. **Source** is where a file is read, and **target** is where a new file is written. Normally, such a command has a **syntax** (grammatical structure) like this:

```
A>copy source target
```

If only one parameter is specified, it is presumed to be the source, such as:

```
A>copy b:sample.txt
```

Here `SAMPLE.TXT` is read from drive B and written to drive A. The omitted second parameter is presumed to be the default drive. The above command is thus the abbreviated version of the one below:

```
A>copy b:sample.txt a:
```

If the source has no drive prefix, it defaults to the default drive, and a target is expected, such as:

```
A>copy sample.txt b:
```

If the target drive (b:) is omitted in this example, DOS assumes that you want to copy a file to itself. You are not allowed to do that.

DOS does not allow you to create a duplicate file of the same name on the same disk. This makes sense; having two files of the same name on the same disk definitely creates confusion. You may be able to tell them apart by file lengths and contents, but DOS cannot; it distinguishes files by their names only. However, if you divide a disk into various directories, you can store an identical copy of the same file in each directory without causing confusion. Directories are covered in Chapter 5.

8. Create a duplicate file with a different name:

```
B>copy sample.txt sample1.txt
        1 File(s) copied
```

There are now two identical files on the same disk under different names, `SAMPLE.TXT` and `SAMPLE1.TXT`. The single-digit difference in the two names enables DOS to treat the files as two distinct entities.

9. Copy a file to another drive and change the name of the target file:

```
B>copy a:sample.txt sample1.txt
```

This copies a file named `SAMPLE.TXT` from drive A (source) to drive B (target) and changes the target file name to `SAMPLE1.TXT`. The original `SAMPLE1.TXT`

in drive B, if any, is replaced with the contents of SAMPLE.TXT from drive A. Notice that the drive prefix in the target is omitted. It could be specified as below and the result would be the same:

```
B>copy a:sample.txt b:sample1.txt
```

10. Show the directory entries in B:

```
B>dir
```

You should see two files of the same date, time, length, but different names:

```
SAMPLE  TXT     268     06-25-90      3:30p
SAMPLE1 TXT     268     06-25-90      3:30p
```

You can now put the write-protect tab back on the DOS-TA disk to prevent inadvertent erasure.

Practice 2-2

- ☐ 1. Copy SAMPLE.TXT from the DOS-TA disk to your practice disk and use MYFILE.DOC as the file name on your disk.
- ☐ 2. Copy MYFILE.DOC to the same disk with the new name of FILE2.TXT.
- ☐ 3. Use TYPE to show FILE2.TXT.
- ☐ 4. Repeat #3, but press Ctrl-C before the entire file is displayed.

Drill

_____ 17. The command A>copy filename b: will copy a file:
   a. from A to B
   b. from B to A
   c. either a or b
   d. nowhere; it's illegal

_____ 18. The command B>copy a: filename b: will copy a file:
   a. from A to B
   b. from B to A
   c. either a or b
   d. nowhere; it's illegal

_____ 19. Which command below is legal?
   a. A>b:type filename
   b. A>a:copy filename b:
   c. A>type filename b:
   d. B>copy b:filename a:

_____ 20. Which command below is *illegal*?

a. `B>dir b:`
b. `A>type b:`
c. `A>copy a b:`
d. `B>dir a: /w`

Renaming Files

✎ **Use REN to change a file's name to a new one. The new name must not match a file already in existence on the same disk.**

A file name can be changed with the REN (RENAME) command. REN, like COPY, is used with two parameters:

```
REN source target
```

Here you give the source (an existing file) a new name. The target file is an exact copy of the source, except with a new name. If the source does not exist or the target already exists, this error message appears:

```
Duplicate file name or File not found
```

With B still the default drive, follow these steps to change file names:

1. Change a file name to that of another existing file:

```
B>ren sample.txt sample1.txt
Duplicate file name or file not found
```

If SAMPLE1.TXT already exists on the same disk, you are not allowed to change SAMPLE.TXT to SAMPLE1.TXT.

2. Change to a different name:

```
B>ren sample.txt sample

B>_
```

The drive is activated and the DOS prompt reappears, but no message is displayed. That means the renaming has been successful.

3. Show the contents of the file:

```
B>type sample
```

The screen should show the same text file as SAMPLE.TXT.

Deleting Files

- ✎ **Use `DEL` (or `ERASE`) to delete a file or group of matching files.**
- ✎ **`DEL` does not actually erase a file from disk; it merely marks the disk space as available for future use.**
- ✎ **Use `DEL` with caution. After a file is deleted, DOS cannot bring it back.**

When you no longer need a file, you can delete it. If a disk runs out of space for storing new files, you can delete some existing files to make room. To delete a file, simply use the DEL or ERASE command.

DEL and ERASE are interchangeable. Either one can be used to remove from disk a file or a group of matching files stored there. DEL is more commonly used because it is shorter.

If you have not changed the setup from the previous section, you are ready to practice deleting files. Follow these steps to delete one file at a time:

1. Delete a file in the default drive:

```
B>del sample.txt
```

Drive B should light up and hum for a while. But no message tells you that one file has been deleted.

2. Use DIR to see whether the file still exists:

```
B>dir sample.txt
File not found
```

The message tells you that DOS can no longer find the file.

3. Go to drive A:

```
B>a:
A>_
```

4. Delete a file in another drive:

```
A>del b:sample
```

5. Use TYPE to see whether a file exists:

```
A>type b:sample
File not found - SAMPLE
```

The two files created by COPY in the previous section are now deleted from your practice disk in drive B. Later in the text, you should use similar steps to delete files created in various practice exercises. This will free disk space for future use. When files you create are to be kept for future use, you will be so instructed.

You can delete all the files on one disk with a single DEL command or delete a group of files matching the specified name. You do so by using DEL with certain wild card characters. These will be explained in Chapter 4.

Warning: Use DEL or ERASE with caution. Once such a command is entered,

DOS goes to work without warning or giving any message. Once a file is deleted with either command, you can no longer access it—unless you have some special programs to recover it. At the present time, no DOS command is capable of recovering a deleted file.

Tips & Tricks

DEL and FORMAT

When you use `DEL` or `ERASE` to delete a file from disk, DOS does not actually erase the file's contents. It merely marks the disk space as available for storing new data.

`FORMAT` is a different story. It wipes the whole disk clean and sets up a completely new system.

There are a number of third-party utility programs that can restore (undelete) a deleted file as long as no new file is saved to the same disk space. The two most popular packages are Norton Utilities and PC Tools. They can also restore files from a hard disk (but not a floppy disk) that has been reformatted. Depending on various factors, restoration may not be complete.

Practice 2-3

- ☐ 1. Rename the `MYFILE.DOC` file on your practice disk `NEWDOC`.
- ☐ 2. With A as the default drive and your practice disk in drive B, delete `NEWDOC` on the practice disk.
- ☐ 3. Go to drive B and delete `FILE2.TXT`.
- ☐ 4. Use `DIR` and `TYPE` to verify the nonexistence of `MYFILE.DOC`, `NEWDOC`, and `FILE2.TXT`.

Drill

_____ 21. Which command below is legal?

a. `A>ren file1 b:file2`
b. `A>ren b:filename`
c. `A>ren b:filename newfile`
d. `A>a:ren file1 file2`

_____ 22. The command `A>del b:filename` will cause DOS to erase from disk all the information related to `FILENAME`. True or false?

_____ 23. Which command will succeed in transferring and renaming a file?

a. `A>copy file b:text`
b. `A>ren file b:text`
c. `A>type file b:text`
d. `A>dir file b:text`

_____ 24. Which command has a required parameter missing?

a. `A>copy filename b:`
b. `B>dir`
c. `A>del filename`
d. `B>ren filename`

Copying a Disk

- **Use `DISKCOPY` to create an exact duplicate of a disk.**
- **`DISKCOPY` can duplicate a DOS system (boot) disk, including the hidden files.**
- **`DISKCOPY` is an external command. You must tell DOS where to access it.**
- **`DISKCOPY` will erase files, if any, on the target disk, and format it if it is unformatted.**

You have learned to use `COPY` to create duplicate files. You can also use the `DISKCOPY` command to create a duplicate disk containing the identical items as the original disk. Before you use this command, it is a good idea to protect the source disk with a write-protect tab. If this disk is inadvertently designated as the target, the files stored there will not be destroyed.

Warning: Keep in mind that DISKCOPY is a destructive command. It will erase everything on the target disk. If the disk is not formatted, DISKCOPY will format it before copying.

DISKCOPY expects two parameters, source and target:

```
A>diskcopy a: b:
```

Here it will copy from drive A to drive B.

DISKCOPY is an external command. That means you need to put the program named DISKCOPY.COM in the default drive or use a path prefix to direct DOS to a drive to access the program.

Find out where the DISKCOPY.COM program is located by using the DIR command (Install disk in version 4 and Operating disk in version 3.3). Insert various disks in drive A and use DIR to see whether it is there. If not, replace another disk and try again. When the file is found, keep the disk there and follow these steps:

1. Enter the command to copy from A to B:

```
A>diskcopy a: b:
Insert SOURCE diskette in drive A:
Insert TARGET diskette in drive B:
Press any key to continue . . .
```

After the program is loaded (put in RAM), the three-line message appears. You are now instructed to insert a source disk in drive A and a target disk in drive B.[5]

2. Take out the disk in drive A and replace it with the DOS-TA disk, which should be write-protected. Insert a formatted blank disk or a new unformatted disk in drive B. If the disk has been formatted and used, make sure it contains no file you want to save. All the files on the target disk will be erased.

3. Press a key to start copying; press Ctrl-C if you want to abort. Copying begins immediately and the message below appears:

```
Copying 40 tracks
9 Sectors/Track, 2 Side(s)
Volume Serial Number is 3D65-18CE
Copy another diskette (Y/N) ?n
```

The above message is based on version 4. Other versions will have no serial number and display slightly different messages.

4. Press N when you are prompted with Y/N (for Yes or No).

You have now created a duplicate of the disk that accompanies this book. Should something happen to the original and you can no longer use it, the duplicate will work just as well. You may want to put a proper label on this new copy and put the original copy in a safe place. Use the new copy as a working copy and the original as a reserve.

[5] If you have a single floppy drive, you can use any command below:

```
A>diskcopy
A>diskcopy a: a:
A>diskcopy a: b:
```

DOS will instruct you to swap disks.

In the future, when you acquire a new program, use the DISKCOPY command to create duplicates, as explained above. If you need to boot DOS with a system disk, you can also use this command to make a copy of the original system disk. DISKCOPY, unlike COPY, is capable of copying hidden files.

COPY, DISKCOPY, and other copying commands will be explained in greater detail in Chapter 6.

Internal and External Commands

- **Internal commands are included in the COMMAND . COM file and loaded to RAM. They can be used regardless of the default drive.**
- **External commands are individual programs stored on a disk. To use an external command, you need to guide DOS to where the program is stored, unless it is in the default drive.**

There are two kinds of DOS commands, internal and external. **Internal commands** can be used anywhere, regardless of the drive you are in. They are included in the COMMAND . COM file, which is loaded to RAM when DOS is booted.

External commands are separate program files stored on different disks in the DOS package. These files have the extension COM or EXE. These programs can be run by entering correct file names on the command line. The extensions are not necessary when you want to run such a program.

External commands are not included in COMMAND . COM. To keep COMMAND . COM small so that it will not take up too much memory, infrequently used DOS commands are intentionally left out. A small operating system is considered a virtue because there is more memory left for application programs and data files.

To use an external command, put the matching program on the disk in the default drive. If DOS cannot find this program to run, it will display an error message:

```
A>diskcopy a: b:
Bad command or file name
```

If the program is not located in the default drive, add a path prefix to tell DOS where to load this program, such as:

```
A>b:diskcopy a: b:
```

This command tells DOS to go to drive B to run the DISKCOPY . COM program and copy from drive A to B.

Hard Disk Users: If you have a hard disk, you can store most or all DOS external commands in a directory named DOS. When you want to run an external command, simply direct DOS to go there by entering a command like this:

```
A>c:\dos\diskcopy
```

This command tells DOS to go to the directory named DOS in drive C to run the DISKCOPY . COM program stored there and copy from drive A to drive A. You will

be prompted to swap disks. Chapter 5 tells you how to use directories; Chapter 6 provides more details related to `DISKCOPY`.

Another way of using an external command is to use the `PATH` command to direct DOS where to search for executable files. This will also be explained in Chapter 5.

If you have only floppy drives or if you are not allowed to use the hard disk in your PC, you have to load a program from a floppy disk every time you need to use an external command. This means you must know what programs are stored on what disk. You can stick a disk in a drive and use DIR to find out. This may be a laborious chore with version 4 since it comes in six 360KB disks. In this situation, a simpler way is to print out a directory list, which allows you to quickly locate a file. This is covered next.

Printing a Screen Display and Disk File

- **Use the Shift-Print Scrn key combination to print what is displayed on the screen.**
- **Use Ctrl-P or Ctrl-Print Scrn to echo every move from the console (keyboard and screen) to the printer.**
- **Use `COPY` to copy a disk file to the printer.**
- **Use the redirection symbol > with `DIR` and `TYPE` to print directory entries and a file's contents.**

There are a number of ways to produce a **printout**—putting text on paper. You can display text lines on the screen and print the screen display. You can echo every keystroke to the printer as well as to the screen. Or you can send a file stored on a disk to the printer.

Printing a Screen Display

Follow these steps to print a screen display:

1. Insert the DOS-TA disk in drive A (or any disk containing the file specified in step 4 below).
2. Turn on the printer and roll in paper to the proper position.
3. Clear the screen:

   ```
   A>cls
   ```

4. Display text on the screen:

   ```
   A>type sample.txt
   ```

5. Print the screen display by pressing the following key combination:

   ```
   A><Shift>-<Print Scrn>
   ```

Here you are to hold down a Shift key and press the Print Scrn key (Prn Scn on an older keyboard); on a newer keyboard, pressing Shift is not necessary. The screen display should be sent to the printer.

If your printer does not print, it may be offline and not accepting messages. If that happens, turn the printer off and on again, and repeat step 5.

Tips & Tricks

Printing Screen Graphics

DOS can print characters displayed on the screen without any trouble. However, when there is a graphics image (which is basically composed of numerous dots), you need to load a program named `GRAPHICS.COM` first. Without this extra step, a blank image will be sent to the printer and the printer will not print anything.

When you are ready to print a screen image (do not do this at this time), follow these steps:

1. Insert in drive A the disk containing `GRAPHICS.COM`; use DIR to find it.
2. Load `GRAPHICS.COM`:
   ```
   A>graphics
   ```
3. Run your application program and display the graphics on the screen.
4. Print the screen display by holding down Shift and pressing the Print Scrn key (pressing Shift is unnecessary on a newer keyboard).

If you want to do some unusual printing, such as printing color graphics, check the `GRAPHICS.TXT` file stored on the Operating 2 disk in version 4. It explains numerous available options.

Simultaneous Displaying and Printing

Another way to get a hard (printed) copy of what is shown on the screen is to echo every step to the printer. The steps below demonstrate how to do it:

1. Get the printer ready.
2. Press Ctrl-P (or Ctrl-Print Scrn)—hold down Ctrl and press either key.
3. Display text on the screen:
   ```
   A>type sample.txt
   ```

On a daisy-wheel printer, whatever you type, including the above command, is displayed on the screen and sent to the printer; the printer produces an exact copy of what is shown on the screen. On a dot-matrix printer, typed characters are not printed until Esc, Enter or Backspace is pressed.

If you do not want a command to be printed, type the command first (without pressing Enter), then press Ctrl-P, and finally press Enter. If you enter the above command by using this method, only the text is printed.

4. Press Enter twice.
5. Type a line (do not press Enter):

```
A>This is a test.
```

6. Delete the line by holding down the Backspace key.
7. Press Ctrl-P to disable the echo effect.
8. Press Enter twice.

This time, the printer no longer responds to what is happening on the screen.

Copying a File to the Printer

If you already have a file containing only ASCII text, you can use `COPY` to print it. Follow these steps:

1. Get the printer ready.
2. Copy a file from disk to the printer:

```
A>copy sample.txt prn
```

`PRN` is a DOS device name. DOS recognizes it as a printer. You can also use `LPT1` to produce the same result:

```
A>copy sample.txt lpt1
```

The text file is now printed. The screen shows only this message:

```
1 File(s) copied
```

Redirecting Output to the Printer

Another technique is to redirect output from the screen to the printer. Whatever normally goes to the screen can be redirected to the printer.

In the previous section, you learned the virtue of printing out a directory list for easy locating of DOS command files. Now follow these steps to do that:

1. Insert a DOS disk (or any disk) in drive A.
2. Send its list of directory entries to the printer, which should be left on:

```
A>dir /w > prn
```

All the directory entries of the disk in drive A will be printed in the wide format. Notice that the screen does not show the output of the command; it is redirected to the printer instead.

3. Replace a disk and repeat step 2 until all the DOS disks are done.

Now, cut up the printout and insert each piece in a disk envelope. In the future, you can just look at each piece of printout to quickly locate an external command. As you will need to access many external DOS commands, this trick can save you a lot of time.

You can also use `TYPE` to print a disk file by entering a command like this:

```
A>type sample.txt > prn
```

This redirects output from the screen to the printer. The text is printed but not displayed. The techniques of redirecting output will be covered more thoroughly in Chapter 8.

Practice 2-4

- ☐ 1. Use `DIR` to display the directory entries of a disk and then print the screen display.
- ☐ 2. Repeat the above, but echo every move to the printer.
- ☐ 3. Use the `TYPE` command to send the `SAMPLE.TXT` file to the screen and printer. Your command should not be printed.
- ☐ 4. Repeat the above exercise without displaying text on the screen. (Hint: Use the redirection symbol >.)
- ☐ 5. Use the `COPY` command to send the `SAMPLE.TXT` file to the printer.

Drill

Each command in questions 25–28 is internal. True or false?

_____ 25. `COPY`

_____ 26. `DISKCOPY`

_____ 27. `GRAPHICS`

_____ 28. `TYPE`

_____ 29. If the `DISKCOPY.COM` file is stored on the disk in drive A (and the `PATH` command has not been used), the `B>diskcopy a: b:` command will work. True or false?

_____ 30. Which command below will NOT print?

a. `A>copy filename prn`
b. `A>dir b: > prn`
c. `A>type filename > prn`
d. `A>ren filename > prn`

Summary and Review

Check each item as you review and understand the following highlights of the chapter.

- ☐ 1. A file is a collection of information that DOS can maneuver. DOS can recognize files by their file name extensions. The four common extensions in a DOS package are `COM` (command), `EXE` (executable), `BAT` (batch), and `SYS` (system).
- ☐ 2. Text files, commonly called ASCII files, usually carry the optional extensions of `TXT` or `DOC`. When you buy a software package, pay attention to a file named README. It usually contains information not available in the printed manual.
- ☐ 3. When DOS is booted, the system prompt automatically displays the letter of the default drive. The default drive is where DOS will look for a file when instructed to do so.
- ☐ 4. If a command is to act on a file not in the default drive, you need to add a parameter after the command, specifying which drive or file the command is to act on.
- ☐ 5. The default drive can be changed. To do so, simply enter a letter followed by a colon.
- ☐ 6. The `DIR` command can display directory information related to a file or an entire disk. You can find out useful information such as file names, sizes, creation dates, etc.
- ☐ 7. `DIR` has two switches, `/P` (pause) and `/W` (wide). A switch is an option that can cause a command to behave in a slightly different manner. The `/P` switch displays a screen and then pauses. The `/W` switch displays file names in the wide format of five columns.
- ☐ 8. When a command such as `DIR` displays more than a screen of information, the display normally continues to scroll until the end is reached. If you want to halt the scrolling at any point, just press Pause. If your keyboard lacks this key, use Ctrl-S or Ctrl-Num Lock.
- ☐ 9. When you want to see what is in a file, use the `TYPE` command to display its contents on the screen. `TYPE` can clearly display all the characters of a pure ASCII file. Other files will display strange characters.
- ☐ 10. You can use `COPY` to create duplicates of individual files. `COPY` expects two parameters, source and target. If only one parameter is provided, DOS treats it as the source and the default drive as the target.
- ☐ 11. `COPY` can copy a file from one disk to another with the same or a different name. It can also make a duplicate file with a different name on the same disk. Two files of the same name, however, cannot be stored on the same disk (in the same directory).

☐ 12. REN allows you to change a file's name. The new name must not be the same as that of another file on the same disk (in the same directory).

☐ 13. Use DEL or ERASE, followed by a file name, to delete a particular file. A deleted file cannot be recovered with any existing DOS command.

☐ 14. Use DISKCOPY to create an exact duplicate of a disk. This command will destroy any file that exists on the target disk and format it if it is unformatted.

☐ 15. An internal command like DEL or REN can be used anywhere. An external command like DISKCOPY, however, must be in the default drive or a path must be provided so DOS can find it.

☐ 16. To create a printout, use one of the following methods:
   a. Press Shift-Print Scrn to print a screen display.
   b. Press Ctrl-P to echo from the console to the printer.
   c. Use the COPY command to copy a file to the printer.
   d. Use > with either DIR or TYPE to redirect output to PRN.

Review Questions

Write an answer for each question below.

1. What is a file? What do these file name extensions mean: COM, EXE, BAT, SYS?
2. What is the significance of a README file?
3. What does the default drive mean? How do you change it?
4. If you use DIR without any parameter, what does it do? What are the items displayed?
5. What are DIR's two switches? What does each do?
6. How can you halt a long screen display?
7. How do the terms switch and parameter differ?
8. How many parameters does COPY expect? What happens if only one is specified?
9. Can you make a duplicate of a file with the same name and on the same disk? Explain.
10. If both A.TXT and B.TXT exist on the same disk (directory), explain the consequences of these two commands:
    a. `A>ren a.txt b.txt`
    b. `A>copy a.txt b.txt`
11. Explain what DISKCOPY will do to a target disk.

12. If drive A contains an original disk provided by a software vendor and drive B contains your own disk, what will happen when you enter the `A>diskcopy b: a:` command?

13. Explain what each key combination does.
    a. Ctrl-P
    b. Ctrl-S
    c. Shift-Print Scrn

14. What does the command `A>dir b: /p > LPT1` do?

15. How does the command `A>copy filename prn` work?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
Disk serial number	Not available	New
Drive failure message	`Abort, Retry, Ignore` (before v. 3.3)	`Abort, Retry, Fail` (v. 3.3 & 4)

Managing Disks

Chapter 3

TOPICS:

Different Types of Storage Media
Floppy Disk Types and Parts
Caring for a Floppy Disk
Analysis of Floppy Disks
Disk Incompatibility
Formatting Floppy Disks
What Formatting Does
Volume Label
Checking a Disk
Setting Up a Hard Disk

COMMANDS COVERED:

`FORMAT`, `VOL`, `LABEL`, `CHKDSK`, `FDISK`

NEW TERMS AND SYMBOLS:

CD-ROM, DAT, write-protect notch/switch, index hole (sector notch), read-write opening, disk spindle, track, sector, boot record, FAT (file allocation table), cluster (allocation unit), volume label, platter, cylinder, partition

REQUIRED MATERIALS:

1. Three unformatted disks (or ready for formatting)
2. Access to `FORMAT.COM`, `LABEL.COM`, `CHKDSK.COM`, `FDISK.EXE`

Quick Start

✎ **To identify different parts of a floppy disk:**
See Figures 3.1 and 3.2.

✎ **To understand how DOS organizes a floppy disk:**
See Figures 3.3 and 3.4.

✎ **To format a disk:**

```
A>format b:
```

✎ **To use FORMAT's various switches:**
See the list on page 71.

✎ **To see a disk's volume label:**

```
A>vol
```

✎ **To see and/or change a disk's volume label:**

```
A>label b:
```

✎ **To check an entire disk:**

```
A>chkdsk b:
```

✎ **To check a file's contiguity:**

```
A>chkdsk b:filename
```

✎ **To check a disk's fragmentation:**

```
A>chkdsk b:*.*
```

✎ **To check a disk and display all the file names:**

```
A>chkdsk b: /v
```

✎ **To recover files due to corrupted FAT:**

```
A>chkdsk b: /f
```

✎ **To recover a file due to bad sectors:**

```
A>recover b:filename
```

✎ **To partition a hard disk:**

```
A>fdisk
```

When you create a file, you may need to save it for later use. There are a wide variety of electronic media on which you can save your files. The most popular are floppy and hard disks.

A number of DOS commands let you manage disks. You can use them to prepare disks for use, check their contents, change their volume labels, and so on.

This chapter explains these closely related topics in this order: (1) different kinds of storage media, (2) different parts of a floppy disk, (3) ways to protect a floppy disk, (4) how DOS organizes a floppy disk, (5) different kinds and sizes of floppy disks, (6) preparing (formatting) a disk for use, (7) analysis of a formatted disk, (8) displaying and changing a disk's volume label, (9) checking the contents of a disk, and (10) preparing a hard disk for use. The emphasis is on understanding and managing floppy disks. You are most likely to use them in your daily work. By knowing how they work and how to maintain them, you are less likely to lose files.

You will encounter quite a few computer numbers in this chapter. If they confuse you, consult the terms and numbers explained in Appendix F.

Different Types of Storage Media

✎ **Many types of storage media are available today. The most common ones are floppy and hard disks.**

A wide variety of media are available to store computer data. As computer technology continues to advance, more variety and higher capacity can be expected.

The most common medium is the floppy disk, or diskette. It is the most versatile and economical way to save data. Every PC must have a least one floppy drive to handle floppy disks. Floppy disks come in various sizes and capacities, as explained below.

Hard disks, also known as fixed disks, are becoming more common. Their prices have fallen as fast as their capacities have increased. As more application programs require larger and faster disk drives, more and more users are finding it hard to compute without a fast and decent-sized hard disk. Only a few years ago, 10MB disks were common. Today you will have difficulty buying a hard disk smaller than 20MB. Hard disks of several hundred megabytes and fast access time of 20 milliseconds (in contrast to 80 ms for earlier models) are increasingly common.

CD-ROMs (Compact Disk–Read Only Memory) are increasingly used to store and distribute massive volumes of data. These disks resemble compact disks and can be handled by a specially equipped disk player. The stored data can be read but not altered. This arrangement is useful for storing and distributing permanent data. Microsoft, for example, markets a package called Bookshelf that contains a host of reference works.

In late 1989, some major manufacturers began to negotiate a standard for a new technology, **DAT** (digital audio tape). A DAT tape, resembling an audio cassette tape, can store a vast amount of data. It can be used like a floppy disk for random access, but is most useful in backing up large files.

Other types of media abound, but none has so far matched the popularity of floppy disks. In this book, we will concentrate on understanding them.

Floppy Disk Types and Parts

✎ **On the surface, a floppy disk consists of a protective jacket, a write-protect device, an indexing device, a spinning device, and an opening for reading and writing.**

There are two kinds of floppy disks in use today: 5.25 inch and 3.5 inch. The former can be divided into 360KB and 1.2MB, and the latter into 720KB and 1.44MB.

5.25-Inch Floppy Disks

The first IBM PC was equipped with a floppy drive that could handle 5.25-inch, single-sided, double-density (SS/DD), eight-sectored (see below for the meanings of these terms) floppy disks. Each disk could store only up to 160KB of data. Double-sided (DS/DD) drives and disks soon became available, making a total disk capacity of 320KB. This disk's storage capacity was finally increased to 360KB by a new DOS version.

The AT brought with it quad-density (IBM refers to it as high capacity; others call it high density or DS/HD) drives and disks; At the same time, Microsoft released DOS 3.0 to accommodate these new disks. They could each store up to 1.2MB of data. Today most AT models come with this floppy drive.

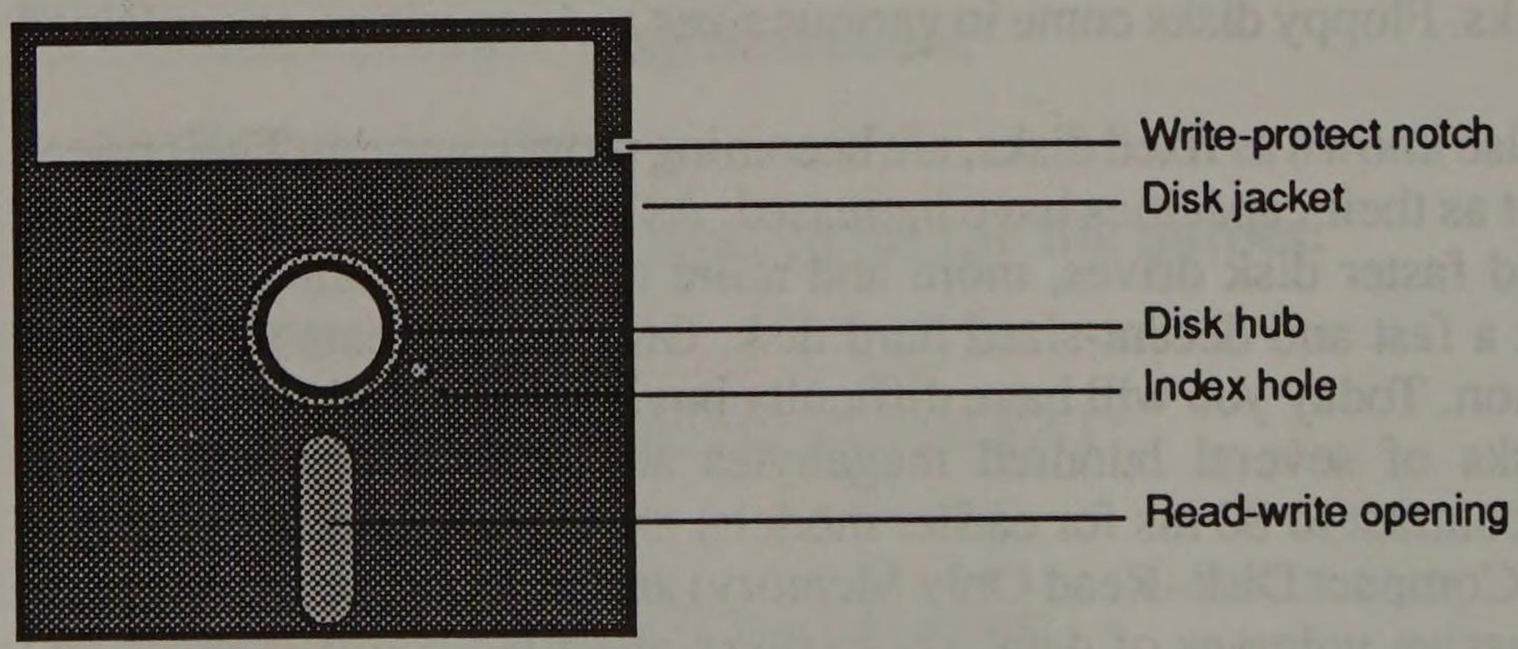


Figure 3.1 A 5.25-inch floppy disk

As shown in Figure 3.1, a 5.25-inch disk has a number of parts you need to be aware of. The **write-protect notch** can be covered with a write-protect tab, which comes with a disk package, to prevent writing to the disk. Such a disk can be read; the stored data, however, cannot be altered. When you want to write to it or delete some files there, just peel off the tab.

The disk hub is a reinforced ring, which a drive can grab to spin the disk inside the disk jacket. As the disk spins, the **index hole** regulates the timing for reading the data or writing to the disk in the **read-write opening**.

3.5-Inch Floppy Disks

The 3.5-inch drive was made popular by the Apple Macintosh. IBM made it the standard for its entire PS/1 and PS/2 lines, thus abandoning the 5.25-inch standard. Microsoft also made available DOS 3.2 to handle this new media. Depending on drives and disks, a 3.5-inch disk can be formatted to either 720KB (double-density) or 1.44MB (quad-density) capacity. Some vendors refer to them as minidisks or microfloppies.

Many of today's compatibles are also equipped with 3.5-inch drives. All the Tandy PCs today, for example, use only 3.5-inch drives. Most vendors, however, offer buyers a full spectrum of options. In some models, you can choose among four drives or some sort of combination.

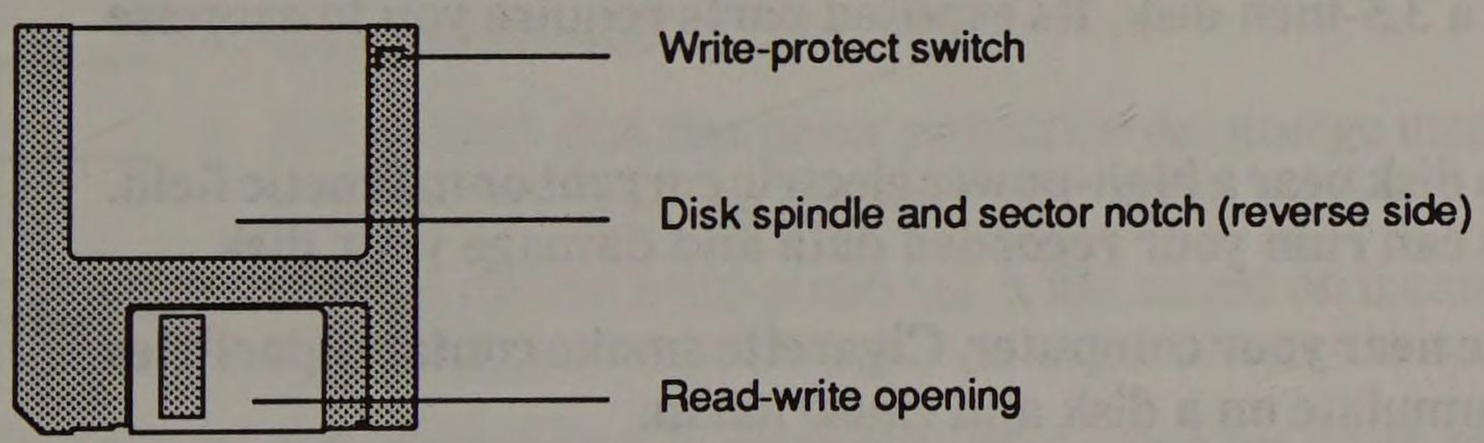


Figure 3.2 A 3.5-inch floppy disk

A 3.5-inch disk's storage media (shown in Figure 3.2), just as floppy and fragile as that of a 5.25-inch disk, is housed in a rigid plastic case and thus much better protected. The **write-protect switch** can be pushed to the edge (showing a see-through hole) to prevent writing to the disk. This is comparable to breaking off an attached plastic piece on an audio cassette tape to prevent recording. When the hole is covered, data can be written to the disk.

The read-write opening is protected with a metal shutter. You can slide the shutter to expose it. When the disk is inserted in a drive, the shutter is automatically opened, thus exposing the disk for reading and writing.

The **disk spindle** serves the same purpose as the disk hub and the **sector notch** as the index hole of a 5.25-inch disk. They control the spinning and timing of reading and writing to the read-write opening.

Although 360KB and 1.2MB disks look exactly alike, 720KB and 1.44MB disks differ in one important aspect. A 1.44MB disk has an extra square hole on the opposite end of the write-protect switch. Some people drill a hole on a 720KB disk so it can be used in a 1.44MB drive. This is not advisable.

The 3.5-inch floppy disk's capacities are expected to increase. In 1990 IBM was reportedly preparing to market a floppy drive that can handle a 4MB, 3.5-inch disk, which can be formatted to 2.88MB of usable disk space. A more exciting offering may come in the form of a 20MB, 3.5-inch floppy drive. It is supposed to be available

sometime in 1990. It reportedly can read and write to existing 3.5-inch 720KB and 1.44MB disks.

Caring for a Floppy Disk

✎ **A floppy disk must be treated with tender loving care.**

A floppy disk operates very much like a video or audio tape. The recording media is a piece of plastic coated with metal oxide. When recording, electric current is sent to magnetize some parts. The magnetized portions can then be read.

The media on which data is written is fragile and needs special protection. If you do not take good care of it, it could become useless and the data written on it could be lost forever. Consequently, you should take the following precautions:

✎ **Do not touch the exposed parts. A 5.25-inch disk is not so well protected as a 3.5-inch disk. Its exposed parts require you to exercise extra care.**

✎ **Do not take a disk near a high-power electric current or magnetic field. These forces can ruin your recorded data and damage your disk.**

✎ **Do not smoke near your computer. Cigarette smoke contains particles that can accumulate on a disk and cause harm.**

✎ **Do not expose a disk to dust. Put it in an envelope or enclosed storage box whenever you can.**

✎ **Do not expose a disk to extreme temperatures or strong light.**

✎ **Do not handle a disk roughly, such as bending, twisting, or putting strong pressure on it.**

✎ **Always back up (duplicate) your important files. A disk can go bad for no apparent reason. For this reason, DOS has provided various commands to back up files, including `COPY`, introduced in Chapter 2.**

The metal oxide on a disk can lose its strength after a long period of heavy use. The disk may no longer record data reliably. It is then time to discard the disk. A disk drive read/write head, like a tape recorder record/play head, can have an excessive accumulation of metal oxide particles, which could render the drive unreliable. There are commercial devices that can clean up the accumulation. If a drive is heavily used, you may need to give it a periodical cleanup.

If you want to protect the data on a disk, you should put a write-protect tab on it. This prevents data from being written to it and reduces the possibility of losing the data already there. PC users, veterans and novices alike, can make simple but disastrous mistakes. For example, you may intend to diskcopy from A to B, but end up instructing DOS to diskcopy from B to A. If disk A contains something you want to save, you may have a disaster on your hands. On such an occasion, a write-protect

tab will prevent the mistake (and could save you from a heart attack). Most 5.25-inch disks are sold with write-protect tabs. If you do not have one handy and are not too sure what a particular command will do, use any kind of adhesive tape to cover the write-protect notch.

Drill

Select a disk capacity in the list to answer questions 1–4:

a. 360KB, b. 720KB, c. 1.2MB, d. 1.44MB.

_____ 1. 3.5-inch double-density.

_____ 2. 3.5-inch quad-density.

_____ 3. 5.25-inch double-density.

_____ 4. 5.25-inch quad-density.

_____ 5. A 5.25-inch disk has better protection on storage media than a 3.5-inch counterpart. True or false?

_____ 6. When a disk is write-protected, a file stored on it can still be deleted. True or false?

_____ 7. A 5.25-inch disk is write-protected when the write-protect notch is covered. True or false?

_____ 8. A 3.5-inch disk is write-protected when the write-protect switch is open (showing a see-through hole). True or false?

_____ 9. To allow a drive to access the storage media of a 3.5-inch disk, you must manually open the read-write opening before inserting the disk in the drive. True or false?

_____ 10. Users of a CD-ROM normally cannot alter the data stored on the disk. True or false?

Analysis of Floppy Disks

✎ **After formatting, a disk is divided into a number of tracks, each of which is in turn divided into a number of sectors. Each sector has 512-byte storage capacity.**

A floppy disk has two sides. Most vendors sell double-sided disks. That means both sides are certified as usable. Today some disks, though not many, are sold as single sided at a lower price. That means that the manufacturer guarantees only one side. However, both sides are useful and are likely to perform equally well.

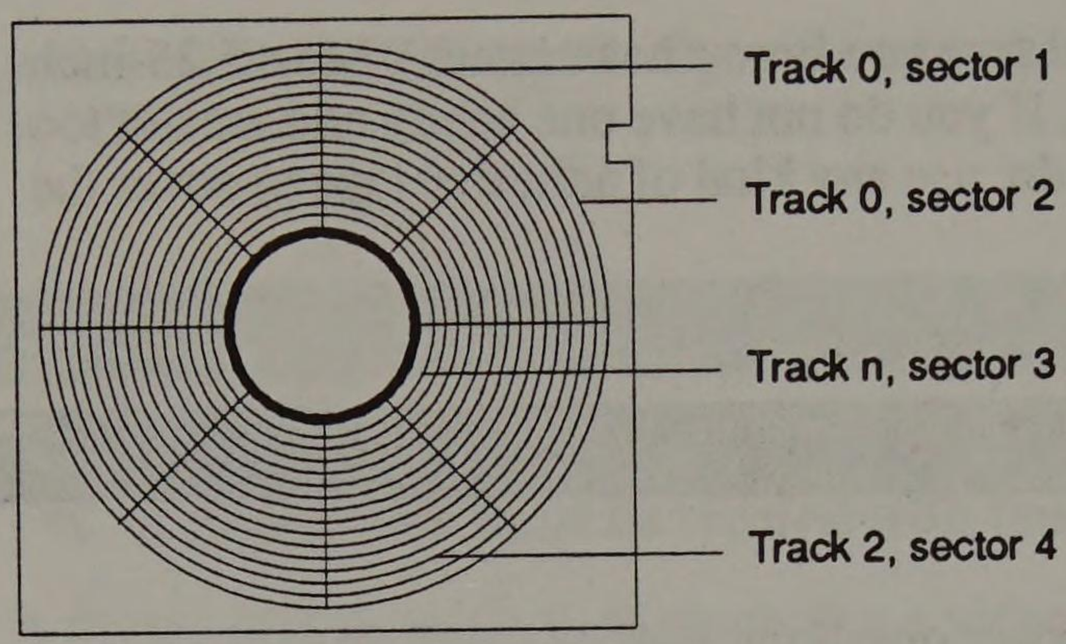


Figure 3.3 Dissection of a formatted disk

To understand how DOS organizes and utilizes a disk, you need to be familiar with these two terms: track and sector. If the protective jacket is taken off, a floppy disk resembles a compact disk in size and shape. This round disk, after formatting (as explained below), is divided into concentric rings called **tracks**, as shown in Figure 3.3. The outermost ring is track 0, and the innermost track 39 on a 5.25-inch double-density disk. A quad-density 5.25-inch disk has 80 tracks (0–79). A 3.5-inch disk also has 80 tracks.

Each track is divided into a number of **sectors**; each sector has a storage capacity of 512 bytes. A 5.25-inch double-density disk can be divided into 8 or 9 sectors per track by using switches available with the FORMAT command. A 5.25-inch quad-density disk can have up to 15 sectors per track. A 3.5-inch disk can be double density or quad density and formatted respectively to 9 and 18 sectors per track, as shown in Table 3.1.

Table 3.1 Floppy Disk Configurations

Size	Tracks per side	Sectors per track	Capacity
5.25"	40	8	320KB
5.25"	40	9	360KB
5.25"	80	15	1.2MB
3.5"	80	9	720KB
3.5"	80	18	1.44MB

A 5.25-inch, double-sided, double-density disk formatted to 9 sectors per track has 360KB capacity, which can be calculated with the following formula:

2 sides × 40 tracks × 9 sectors × 512 bytes = 368640

That number is the same as 360 multiplied by 1024, which is 1KB or 2 to the power of 10. (If you have trouble understanding these numbers, see Appendix F for an explanation.)

A double-sided disk drive has two read-write heads, which can handle both sides of a double-sided disk. When DOS writes to such a disk, the first free sector of the first track (track 0) of the first side (side 0) is filled; then the second sector, and so on. When the first track of side 0 is completely filled, DOS goes to the other side (side 1) and follows the same pattern. After that, the second track of side 0 is filled, and so on.

Disk Incompatibility

- **The 5.25-inch and 3.5-inch drives/disks are incompatible.**
- **There is some degree of compatibility between 360KB and 1.2MB systems, and between 720KB and 1.44MB systems.**

Normally you need to match a floppy disk to the correct floppy drive. There are, however, some exceptions to this rule. When you buy a floppy disk, you need to know what kind of drive you have. First, you need to know whether the drive is 3.5 inch or 5.25 inch. The two are totally incompatible. A 3.5-inch disk cannot be inserted into a 5.25-inch drive, and vice versa. Second, you need to know whether your drive is double or quad density. You need to buy disks matching the drive's capacity. These disks are specifically labeled as double density (360KB for 5.25 inch or 720KB for 3.5 inch) or high (quad) density (1.2MB for 5.25 inch or 1.44MB for 3.5 inch).

If you use a variety of floppy drives, double density at home and quad density in the office, for example, here are some more observations you need to keep in mind:

- **A 1.2MB (or 1.44MB) drive can read and write to a 360KB (or 720KB) disk. A 360KB (or 720KB) drive, however, cannot read or write to a disk formatted to 1.2MB (or 1.44MB).**
- **A 1.2MB drive can format a double-density disk to 360KB or 1.2MB capacity. In the latter case, not all the disk space is available for use. Depending on drives and disks, as much as half of the total 1.2MB capacity may be marked as "bad sectors" and thus unusable. The rest may not be too reliable.**
- **A 1.44MB drive can format a double-density disk to 720KB capacity. Some 1.44MB drives can format a double-density disk to 1.44MB capacity, but most cannot. The absence of an extra hole on a 720KB disk usually prevents its formatting to 1.44MB capacity.**
- **A 360KB disk formatted in a 360KB drive can be reliably read and written to by a 1.2MB drive.**

✎ **A double-density disk formatted to 360KB capacity in a 1.2MB drive may not be reliably read or written to by a 360KB drive. This problem is more evident in earlier drives; you are not likely to encounter any problem in later models. A 3.5-inch disk formatted to 720KB in a 1.44MB drive can be handled by a 720KB drive without any problem.**

Finally, you need to be aware of the fact that not all DOS versions can handle recently available drives and disks. The earlier DOS versions were not designed for devices that did not then exist. DOS versions capable of handling a specific drive/disk are shown in Table 3.2.

Table 3.2 DOS Versions and Disk Capacities

DOS Version	Disk Capacity
2.0	360KB
3.0	1.2MB
3.2	720KB
3.3	1.44MB

If you use version 3.3 or later, you need not worry about DOS's capability. It can handle all four types of drives/disks currently in use.[1]

Drill

_____ 11. After formatting, a disk is divided into a number of concentric rings called _____.

_____ 12. Each ring is divided into a number of subunits called _____.

_____ 13. Each of the above units has a storage capacity of ____ bytes.

_____ 14. A double-density drive cannot format a quad-density disk to its full capacity. True or false?

_____ 15. All the DOS versions since 3.0 can handle 3.5-inch drives and disks. True or false?

[1] If you intend to buy a floppy drive to attach to your current system, you must know whether your BIOS and disk controller card can handle it. If you bought an AT-class PC in 1990, it can handle all four types of drives. Earlier ATs and XT-class PCs may not be able to handle newer drives.

Formatting Floppy Disks

- **Use `FORMAT` to prepare a disk for use by DOS.**
- **Specify the /S switch to create a boot (system) disk.**
- **Use a variety of available switches if you want to format disks for use in incompatible drives.**

Warning: Use the `FORMAT` command with great caution. Formatting a disk destroys all the existing data on that disk.

Before a floppy disk can be used by DOS, it must be prepared with the `FORMAT` command. By formatting a disk, DOS divides it into tracks and sectors. In addition, DOS also stores in certain sectors information used to manage the disk. This information enables DOS to find files to read and available space to store data.

Using No Switch

If you use `FORMAT` without specifying a switch, a disk is formatted to the default capacity of the drive. Consider the following:

```
A>format b:
```

This command tells DOS to go to drive A to load `FORMAT.COM` and format the disk in drive B. Keep in mind that `FORMAT` is an external command and, as explained in Chapter 2, you need to tell DOS where to access it, unless it is in the default drive.

Since no switch is specified, the above command will format the disk in drive B according to drive B's capacity. If it is a 5.25-inch double-density drive, the disk will be formatted to 360KB capacity. If it is a 5.25-inch quad-density drive, you will get 1.2MB capacity, and so on.

If you have only one kind of floppy drive, you do not need to worry about `FORMAT`'s complicated switches. By not using any switch, your disks will be formatted to their full capacity.

FORMAT's Switches

The `FORMAT` command has these switches:

`/1`	1 side only
`/4`	Format 360K in a 1.2MB drive
`/8`	8 sectors per track
`/B`	Blank space for later transfer of system files
`/F:n`	Disk capacity, v.4 only
`/N:n`	Number of sectors per track, used with /T
`/S`	System files transferred
`/T:n`	Tracks per side, used with /N
`/V:l`	Volume label; in v.4, if omitted, you'll be prompted

If you have different kinds of floppy drives, such as 1.2MB and 360KB, and need to use a disk in both drives, you may want to pay closer attention to these switches.

If you have a 1.2MB drive, you can format a double-density disk to 360KB capacity. Such a disk can be used by both 360KB drives and 1.2MB drives. To format a double-density disk to 360KB capacity in a 1.2MB drive, try this command:

```
A>format b: /4
```

The /F Switch

Version Variances: The /F switch discussed in this section is available only in version 4, not those before. If you have version 4, you can use the /F switch for most situations. If you use this switch, many other switches are unnecessary. These are the available values you can specify with the /F switch:

```
160, 180, 320, 360, 720, 1.2 (or 1200), 1.44 (or 1440)
```

These numbers are the capacities you want to format your disk to: 160KB (single sided, double density, 8 sectors); 180KB (single sided, double density, 9 sectors); 320KB (double sided, double density, 8 sectors); 360KB (double sided, double density, 9 sectors); 720KB (3.5 inch, double density); 1.2MB (5.25 inch, quad density); and 1.44MB (3.5 inch, quad density).

Thus, instead of using the /4 switch as shown in the preceding example, you could use this instead:

```
A>format b: /f:360
```

This will format a disk in a 1.2MB drive to the capacity of 360KB.

The /N and /T Switches

If you use version 3.3 to handle a 3.5-inch disk, you need to use the /N and /T switches since it does not have the /F switch. To format a 720KB disk in a quad-density drive, try this command:

```
A>format b: /n:9 /t:80
```

This tells DOS to format by 9 sectors per track and 80 tracks each side. If you use version 3.3, the /T switch is not necessary; DOS will default to 80 tracks. Version 4, however, requires both switches.

Formatting to a Wrong Capacity

What happens if you format a double-density disk in a quad-density drive without specifying any switch? With a 5.25-inch disk, formatting will proceed and about half of the 1.2MB capacity will be identified as unusable; the rest is usable but may not be reliable.

You cannot format a 3.5-inch, double-density (720KB) disk to 1.44MB capacity. DOS simply will not let you do that. If you format a 720KB disk in a 1.44MB drive without specifying a switch, this message is displayed:

```
Invalid media or Track 0 bad—disk unusable
Format terminated
Format another (Y/N)?
```

The /S Switch

If you format with the /S switch, you ask DOS to transfer the system to the formatted disk. Two hidden files and COMMAND.COM will be transferred. This disk can then be used for booting DOS and may be called a "boot disk." This switch can be used with any other except /B.

Tips & Tricks

What Is on a Boot Disk?

When you specify the /S switch with FORMAT, DOS transfers the following three system files to the newly formatted disk:

MS-DOS	PC-DOS
IO.SYS	IBMBIO.COM
MSDOS.SYS	IBMDOS.COM
COMMAND.COM	COMMAND.COM

The first two are hidden and the third is not. Most DOS commands, such as COPY, DEL, DIR, and TYPE, cannot "see" these hidden files. Only CHKDSK, DISKCOPY, and SYS can recognize them.

Using the /S switch with FORMAT is not the only way to make a disk bootable. After you format a disk without using any switch, you can use SYS to transfer the hidden system files to the new disk. This will make the disk bootable. See Chapter 7 for more details.

The /B Switch

The /B switch is used to set aside blank space for later transfer of the system files. This reserves enough space in the beginning sectors for two hidden files and COMMAND.COM. You can then save the programs you have developed. This disk can then be marketed without copyright infringement. If a buyer uses the SYS command (explained in Chapter 7) to transfer the system files to this disk, it becomes bootable.

The /V Switch

The /V switch is for volume label. If you use version 3.3 or earlier and specify this switch, DOS will prompt you to enter a volume label after formatting is complete. If /V is not specified, no prompt for volume label appears.

Version 4 will prompt you to enter a volume label after a disk is formatted—even though the /V switch is not specified. If you use the /V switch followed by a colon and a name, this name is used and no prompt appears.

Formatting Steps

Now that you know how to format floppy disks, you should format a few for your future use. Format one with the `/S` switch to be used as a boot disk and two more without this switch just to store files.

Before you proceed further, remember this fact: Formatting destroys all the data stored on a disk. This is no problem with an unformatted disk. But if a disk contains files you do not want to lose, you must use `COPY` to copy them to another disk before formatting the disk.

Now follow these steps to format a new or used disk:

1. Insert a disk containing `FORMAT.COM` in drive A and a disk to be formatted in drive B. If you would rather be safe than sorry, write-protect the disk in drive A.[2]

2. Format without transferring the system files:

```
A>format b:
Insert new diskette for drive B:
and press ENTER when ready...
```

3. Press the Enter key to begin. The screen begins to display a running number, showing the percentage of the disk formatted. When done, this message appears and you are prompted to enter a volume label:

```
Format complete
Volume label (11 characters, ENTER for none)? disk1
```

4. Enter a label (`DISK1`), and this message appears:

```
362496 bytes total disk space
362496 bytes available on disk
  1024 bytes in each allocation unit
   354 allocation units available on disk
Volume Serial Number is 0C48-19D2
Format another (Y/N)?
```

5. Press `Y` to format another disk.

6. Insert a new disk in drive B and repeat steps 3 and 4.

7. At step 4, press `N` to discontinue.

If you want to use `FORMAT` with a different switch, you must end it and then enter it with another switch.

8. Replace the disk in drive B and enter this command:

```
A>format b: /s
```

9. Repeat steps 3 and 4. After step 4, press `N` to end.

If you have followed the above steps, you will have formatted two disks without the system files and one with the system files for booting DOS. From now on you can use this boot disk to start DOS and the other two to store files. Whenever you need to format more disks, just follow the above steps with or without the /S switch.

[2] If you have only one floppy drive, just follow these steps. DOS will prompt you to swap disks at the proper time.

Version Variances

If you have a 360KB drive and version 4, the screen should show this after you finish formatting with /S:

```
A>format b: /s
Insert new diskette for drive B:
and press ENTER when ready...

Format complete
System transferred
Volume label (11 characters, ENTER for none)? sysdisk

    362496 bytes total disk space
    109568 bytes used by system
    252928 bytes available on disk

      1024 bytes in each allocation unit
       247 allocation units available on disk

Volume Serial Number is 351A-1BE7

Format another (Y/N)?n
```

If you have version 3.3 and format a 360KB disk in a 360KB drive without /S or /V, this is the result:

```
A>format b:
Insert new diskette for drive B:
and strike ENTER when ready

Format complete

    362496 bytes total disk space
    362496 bytes available on disk

Format another (Y/Y)?n
```

Before displaying "Format complete," the screen continues to display in version 4 a percent figure and in version 3.3 two running numbers, one for Head (0 or 1) and the other for Cylinder (0 through 39).

In version 3.3, unless you specify the /V switch, you will not be prompted to enter a volume label.

Notice that version 3.3 has no serial number as in version 4. This unique hexadecimal number in version 4 is randomly generated by DOS without your doing anything. Furthermore, version 3.3 gives no information related to "allocation units." This will be explained in the next section.

If you have sharp eyes and a keen mind, you will have noticed that some numbers do not add up. For example, 360KB should be 368640 (360 × 1024) but a 360KB disk is formatted to only 362496 bytes, which is only 354KB (354 × 1024). Where are the missing 6KB? They are used by DOS to manage the disk, as explained below.

What Formatting Does

- **Formatting creates a boot record, two file allocation tables, and a directory.**

Formatting a disk does more than what the numbers displayed in the preceding section tell you. Consider these acts DOS performs in doing a simple formatting:

- **Divides a disk into tracks and sectors depending on drive and disk capacity and FORMAT switch(es) specified.**
- **Makes the first sector the boot record.**
- **Allocates the next few sectors for a file allocation table (FAT).**
- **Allocates the next few sectors for a duplicate FAT.**
- **Allocates a number of sectors for a directory.**

The first act was explained in the preceding sections. The others are explained in this section.

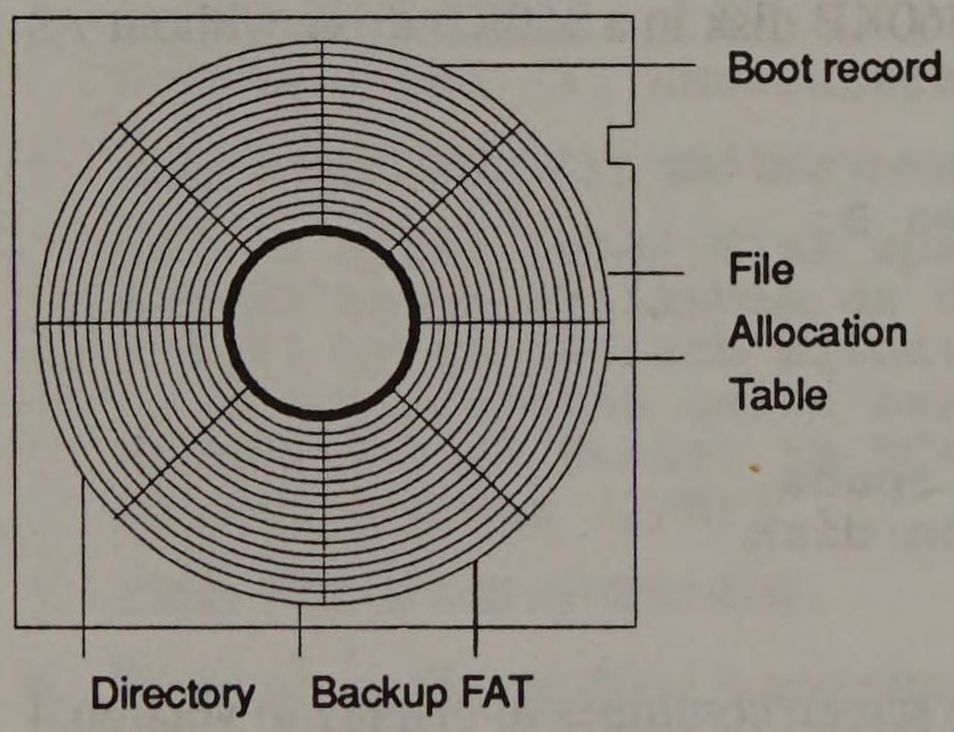


Figure 3.4 Items created by formatting

Boot Record

The **boot record** contains information about the disk—type, size, capacity, etc. The information enables DOS to handle each disk correctly during reading and writing. In addition, there is a short program in charge of loading DOS into memory. If it can find the system files on the same disk to load, control is passed to them. If not, this error message stored in the boot record is displayed:

```
Non-System disk or disk error
Replace and strike any key when ready
```

File Allocation Table

The **FAT** (file allocation table) is a crucial part of a disk. As the name implies, this is a table used for allocating disk space to store files and for accessing the stored

files. DOS uses it to keep track of used, unused, and damaged (unusable) clusters (see below). When you read or write to a disk, DOS depends on the information stored in the FAT to carry out your commands. Without this table, DOS would not know where to find a file to read or available space to write to. Since the FAT is so important, DOS creates a backup FAT in the formatting process and keeps it up to date as files are saved or deleted. Should the original be corrupted, the backup is used. The number of sectors set aside for each table varies with disk capacity: 2 for 360KB, 3 for 720KB, 7 for 1.2MB, and 9 for 1.44MB. The larger a disk's capacity, the more space is needed to manage it.

Directory

After the boot record and two FATs, DOS creates a directory structure to manage file entries. The number of sectors used depends on the type of disk, as shown in Table 3.3.

Table 3.3 Root Directory Sizes and File Limits

Disk Type	Sectors	Maximum Number of File Entries
Single sided	4	64
Double sided	7	112
Quad density	14	224
Hard disk	32	512

DOS requires 32 bytes for managing each file stored on disk. Information related to file name, date-time, file size, and file attributes is stored there. Using the 32-byte number and Table 3.3, we can calculate how many files can be stored in the root directory of a disk. Below is the formula for a double-sided, double-density disk:

7 sectors × 512 bytes per sector / 32 bytes = 112 entries

That means you cannot store more than 112 files (including subdirectories and the volume label, if any) in the root directory of such a 360KB disk. This limit, however, does not apply to the number of files stored in a subdirectory. Directories and subdirectories are explained in Chapter 5.

Cluster

If you use version 4, after completing formatting, DOS displays something called an "allocation unit," as shown below for a 360KB disk:

```
1024 bytes in each allocation unit
354 allocation units available on disk
```

This allocation unit is commonly referred to as a **cluster**. A cluster is the smallest space DOS can work with. A file, even as short as one byte, takes up a cluster of disk space.

How much disk space is a cluster? It varies. On a 1.2MB disk, a cluster is 512 bytes, or 1 sector. On a 360KB disk, it is 1KB, or 2 sectors. On a hard disk, it may be 2, 4, 8, or 16 sectors.

Every time you create a file or directory, a cluster is allocated. On a 360KB disk, for example, a 1-byte file takes up 1024 bytes of space. If the file is at least 1 byte longer than a cluster, the next entire cluster is also allocated. This mechanism is very wasteful of disk space.

Drill

Select a `FORMAT` switch from the list to answer questions 16–20:

a. `/B`, b. `/F`, c. `/N`, d. `/S`, e. `/T`.

_____ 16. Determines the number of tracks.

_____ 17. Determines the number of sectors.

_____ 18. Determines the number of bytes.

_____ 19. Transfers system.

_____ 20. Reserves space for system.

_____ 21. If the `FORMAT` command is used without any switch, it will default to the drive's full capacity. True or false?

_____ 22. The boot record of a disk is used to start the booting DOS. True or false?

_____ 23. A cluster is always 1KB. True or false?

_____ 24. As long as there is disk space, there is no limit to the number of files that can be stored in the root directory of a disk. True or false?

_____ 25. Every formatted disk has two file allocation tables. True or false?

Volume Label

- **Use `VOL` to display a disk's volume label.**
- **Use `LABEL` to display as well as change a disk's volume label.**

A **volume label** is a name you assign to a disk. It can be up to 11 characters long. Since you cannot easily see what data is stored on a disk, you should give it a meaningful name for easier identification.

If you entered a volume label during the formatting process, you can use the VOL (Volume) command to display it and the LABEL command to display and, if necessary, change it.

The VOL command is internal. That means it can be used no matter what the default drive is. To display the volume label of the disk in the default drive, simply enter this command:

```
A>vol

Volume in drive A is DISK1
Volume Serial Number is 0C48-19D2
```

The volume label that you previously entered is displayed in uppercase. If you use version 4, a serial number is also shown.

To display the volume label of another drive, specify the target drive after the command:

```
A>vol b:
```

If you want to give a disk a volume label or change an existing one, you can use the LABEL command. This is an external command and you must tell DOS where to access it. In version 4, the LABEL.COM program is located on the Operating 1 disk. Insert this disk (or any disk containing this program) in drive A and a formatted disk in drive B, and follow these steps:

1. Show the volume label of the disk in drive B:

```
A>label b:
Volume in drive B is DISK1
Volume Serial Number is 0C48-19D2
Volume label (11 characters, ENTER for none)?_
```

The previously entered volume label is displayed, plus a serial number for version 4. You are now prompted for a response. If you enter a new name after the question mark, it will become this disk's volume label and replace the existing one, if any.

2. Press Enter:

```
Delete current volume label (Y/N)?_
```

Since your disk has a volume label, if you just press Enter, you are prompted with Y or N to delete it.

3. Press N:

```
Delete current volume label (Y/N)?n
```

If you respond with N, the original volume label is kept and the DOS prompt is displayed.

4. Change the volume label in drive B:

```
A>label b:disk2
A>_
```

This is a shortcut to changing a disk's volume label. The original DISK1 name is changed to DISK2.

5. Go to drive B:

```
A>b:
B>_
```

6. Display the default drive's volume label:

```
B>vol
Volume in drive B is DISK2
Volume Serial Number is 0C48-19D2
```

The volume label is changed, but the serial number, if any, remains the same.

7. Restore the original name:

```
B>a:label disk1
B>_
```

This command goes to drive A to load LABEL.EXE and changes the default drive's volume label. Remember that the LABEL command is external. That is why we need the drive prefix (A:) to direct DOS to run this program. Since the target is not preceded with any drive prefix, the disk in the default drive (B) is acted on.

Practice 3-1

- ☐ 1. With A as the default drive, use VOL to show the volume label of the disk in drive A.
- ☐ 2. With A as the default drive, use VOL to show the volume label of the disk in drive B.
- ☐ 3. Use LABEL to give a disk a volume label if it doesn't have one. If the disk already has a volume label, use LABEL to display it without change.

Checking a Disk

- ✎ **Use CHKDSK with a drive as a parameter to check the disk's usage.**
- ✎ **Use CHKDSK with a file name as a parameter to check the contiguity or fragmentation of the file's storage.**

The external CHKDSK (Check Disk) command is very important in managing both hard and floppy disks. It can display the following information:

- ✎ **Volume label, if any, and date of creation**
- ✎ **Serial number for version 4**

- ✎ **Number of files, both hidden and not hidden**
- ✎ **Number of bad sectors, if any**
- ✎ **Amount (in bytes) of available disk space**
- ✎ **Size (in bytes) of each cluster (allocation unit), v.4**
- ✎ **Number of clusters, total and available, v.4**
- ✎ **RAM, total and available**
- ✎ **File names and directories**
- ✎ **Contiguity or fragmentation of files**

The CHKDSK command can be used in a variety of ways. You can:

- ✎ **Check an entire disk**
- ✎ **Check a single file**
- ✎ **Check all the matching files**
- ✎ **Specify either the /F or /V switch**

Checking an Entire Disk

Now insert in drive A a disk containing CHKDSK.COM (the CHKDSK.COM program is stored on Operating 1 disk in version 4) and in drive B a disk to be checked, and enter this command:

```
A>chkdsk b:

Volume SYSDISK     created 10-20-1989 7:15p
Volume Serial Number is 351A-1BE7

    362496 bytes total disk space
     71680 bytes in 3 hidden files
     37888 bytes in 1 user files
    252928 bytes available on disk

      1024 bytes in each allocation unit
       354 total allocation units on disk
       247 available allocation units on disk

    655360 total bytes memory
    515648 bytes free
```

There are four sections in this display. The first shows volume label and serial number. If you have not entered a volume label, neither it nor the creation date will be available. If a disk is formatted by version 3.3 or earlier, the serial number is not available.

The creation date is determined by the latest time you entered the volume label. If you use the LABEL command to change it, the creation date will be based on that.

If you use LABEL to delete a volume label, neither the label nor the creation date is available.

The second section contains a series of numbers related to the disk. The first is the total disk capacity. The second and third show respectively the number of hidden and unhidden files. The number for hidden files includes the volume label. If you format a disk with the /S switch but without a volume label, the number of hidden files will be 2 instead of 3 as shown in our display. The same is true if you use LABEL to delete an existing volume label.

The number for available disk space is probably the most important. It tells you how many bytes of disk space are left. If there is little or no space left, it is time to use another disk or delete some files.

The third section shows cluster (allocation unit) information. As explained earlier, a cluster can be one, two, or more sectors, depending on what kind of drive/disk is involved. These numbers are not available if you use version 3.3 or earlier.

Finally, the last group of numbers show total installed RAM (or the amount DOS can use) and the amount still available. If your PC has more than 640KB (640 × 1024 = 655360), only 640KB is shown. As more programs are loaded and remain in memory, the amount of free bytes decreases. Before you load another program, you may need to see whether there is enough memory for it.

Checking Matching Files

Instead of checking an entire disk, you can check a single file or all the files matching a certain pattern. To check a single file, enter this:

```
A>chkdsk filename
```

This command goes to the default drive (A) to check a file named FILENAME also on the same disk. All the numbers as displayed earlier are shown, plus this:

```
All specified file(s) are contiguous
```

This message tells you that all the contents of the file are connected and not scattered in various unconnected clusters of the disk.

If you want to check a series of files matching a pattern, try a command like this:

```
A>chkdsk file?
```

If there are files that are not contiguous, they are displayed as shown below.

Checking Disk Fragmentation

As DOS stores data to a disk, it writes to available space. If an adjacent area is used by another file, an expanded file is scattered in various unconnected clusters. If you work with several files and keep expanding them, they will all be divided. To find out whether all the files are contiguous or not, try this command:

```
A>chkdsk b:*.*

Volume DOSBOOK      created 11-11-1989 7:05p
. . .

B:\DIR.05 Contains 3 non-contiguous blocks
B:\TOUR.02 Contains 4 non-contiguous blocks
B:\DISK.03 Contains 5 non-contiguous blocks
```

All the files that are noncontiguous are shown. The contiguous ones are not displayed.

The *.* notation means all the files in the specified drive. You can use the wild card characters of * and ? to find matching files. These characters will be explained in Chapter 4.

The /F and /V Switches

CHKDSK has two switches, as shown below:

```
/F      Fix files; repair damaged files
/V      Verbose; show files and paths
```

The /F switch is used to reconstruct files that have become inaccessible due to a corrupted FAT. The data stays on the disk, but DOS can no longer reach it. If this happens, CHKDSK will report the number and prompt you for a response:

```
5 lost clusters found in 5 chains.
Convert lost chains to files (Y/N)?
```

If you press Y, DOS goes through the motions of rescuing these files without actually doing it—unless you specify the /F switch, like this:

```
A>chkdsk b: /f
```

If you specify this switch and respond with Y, DOS will gather these lost files and save them with the name of FILEnnnn.CHK, with nnnn as four-digit sequential numbers of 0000, 0001, 0002, etc.

When you see these files, you can use the TYPE command to look into their contents. These are ASCII files free of formatting characters. You can retrieve them into your application program and perhaps salvage them. (You may also find garbage, which can be deleted.)

The /V switch displays a verbose (detailed) version, including file names and what directories they are stored in, as shown below:

```
A>chkdsk b: /v

Volume DOSBOOK      created 11-11-1989 7:05p
Directory B:\
   B:\DIR.05
   B:\TOUR.02
   B:\DISK.03
   .   .   .

362496 bytes total disk space
265216 bytes in 8 user files
 97280 bytes available on disk

  1024 bytes in each allocation unit
   354 total allocation units on disk
    95 available allocation units on disk

655360 total bytes memory
515648 bytes free
```

Tips & Tricks

Recovering a File

When you encounter a message like "`Disk read error on drive A`," you can try to recover a file with the external command of `RECOVER`, like this:

```
A>recover b:filename
```

DOS will mark off bad sectors (and store the information in the FAT) and reconstruct the file with readable data. The new file may not be complete, but DOS should be able to read it.

If all the files on a disk are no longer readable, try this:

```
A>recover b:
```

DOS will recover the files in drive B and save recovered data to the same disk in a series of files named `FILEnnnn.REC`, where `nnnn` is a number starting with 0000. This command destroys the disk's entire directory structure; use it only as a last resort.

Practice 3-2

- ☐ 1. With `CHKDSK` in drive A, use it to check drive B without showing files.
- ☐ 2. Repeat the above, except show all the files stored on that disk.
- ☐ 3. With `CHKDSK` in drive A, check the contiguity of all the files in drive A, showing only fragmented files' names.

Setting Up a Hard Disk

✎ **Set up a hard disk by partitioning it with `FDISK` and formatting (using `FORMAT`) it with the `/S` switch.**

Hard Disk Users: This section explains hard disks. It should not concern you unless you have your own hard disk and want to prepare it for data storage.

Warning: Do not use either `FDISK` or `FORMAT` unless you intend to set up a hard disk. Both commands, once implemented, destroy any existing data stored on a hard disk.

There are few occasions when you need to set up a hard disk so it can store data. If you use a PC in a lab, the installed hard disk, if any, is already set up and there is no need for you to worry how it works. If you buy a hard disk, the vendor may have it set up and ready for you to use; if not, you need to do it yourself. And if your hard disk is completely messed up, you will have to set it up all over again. Knowing how to set up a hard disk and how to manage it will enhance your productivity with your PC. This section tells you how to set it up, and Chapter 5 explains how to use directories to make efficient use of it.

Inside a hard disk drive, there may be a number of disks called platters. A **platter** is the equivalent of a floppy disk, but more rigid. Both sides of a platter are also coated with metal oxide like a floppy. By stacking up platters, a hard disk can increase its capacity.

DOS manages a hard disk similar to the way it does a floppy disk. Each side of a platter is divided into a number of tracks called **cylinders** and cylinders are divided into sectors. Typically, a 40MB disk has three platters. Each platter has two sides. Each side has 820 cylinders. Each cylinder has 17 sectors. Each sector, as you already know, has 512 bytes. By multiplying these numbers, you get the total of slightly more than 40MB:

$$3 \times 2 \times 820 \times 17 \times 512 = 42823680$$

Like a floppy disk, a hard disk needs to be formatted before DOS can use it. Before formatting it, however, you need to create one or more units called **partitions**. These two acts call for two DOS commands, `FDISK` (Fixed Disk) and `FORMAT`.

When you want to partition a hard disk, insert a disk containing the `FDISK.EXE` program (the Install disk in version 4) in drive A and enter this command:

```
A>fdisk
```

After the program is loaded, a message like this appears:

```
                MS-DOS Version 4.01
             Fixed Disk Setup Program
        (C)Copyright Microsoft Corp. 1983, 1988

                      FDISK Options

Current fixed disk drive: 1

Choose one of the following:
1.  Create DOS Partition or Logical DOS Drive
2.  Set active partition
3.  Delete DOS Partition or Logical DOS Drive
4.  Display partition information

Enter choice: [1]

Press Esc to exit FDISK
```

If you have another hard disk installed, a fifth option will appear to let you select another hard disk to partition.

If you wish to terminate, just press Esc, and the DOS prompt will reappear.

To see how your hard disk has been partitioned, press 4 and Enter. A display like this may appear:

```
                Display Partition Information

Current fixed disk drive: 1
Partition Status  Type   Size in Mbytes    % of Disk Used
 C: 1        A   PRI DOS         40            99%

Total disk space is    41 Mbytes (1 Mbyte = 1048576 bytes)

Press Esc to return to FDISK Options
```

The above display shows that a 41MB hard disk has been partitioned into one drive as the primary DOS partition and designated as the active partition. That means DOS can be booted from this drive if you turn on the power while leaving the drive A door open.

When you use `FDISK`, you are prompted with a series of menus, each of which has a default selection. The first menu, for example, has 1 as the default option. If you want to choose the default option at each step, just press Enter.

If you choose all the default options, DOS will create one drive out of your hard disk. The size of this drive is based on the hard disk's capacity. Version 3.3 or earlier cannot handle a drive larger than 32 MB; it manages a large disk by dividing it into two or more smaller drives. Version 4 can accommodate a drive as large as 2 gigabytes.

If you want to partition your disk into several drives, when prompted you can give a megabyte or percentage number for each drive.

After `FDISK` runs its course and completes creating disk partition(s), this message appears:

```
System will now restart

Insert DOS Install diskette in drive A:
Press any key when ready . . .
```

Now, you need to format the newly partitioned disk so DOS can use it. If you want to boot from the hard disk, specify the `/S` switch. If you use version 4 and the Install disk is still in drive A, enter this command to complete the job:

```
A>format c: /s
```

Before formatting starts, this warning message appears first:

```
WARNING, ALL DATA ON NON-REMOVABLE DISK

DRIVE C: WILL BE LOST! Proceed with Format (Y/N)?
```

If you enter `Y`, formatting proceeds as when a floppy disk is involved.

If you purchase a hard disk that is 40MB or larger in capacity, it is usually bundled with a different disk-management program. In that case, use it instead of `FDISK` to partition your hard disk and then use `FORMAT` to format it. Such a program is considered by many to be superior to `FDISK`.

Drill

_____ 26. `LABEL` but not `VOL` can change a volume label. True or false?

_____ 27. `CHKDSK` shows available RAM as well as disk space. True or false?

_____ 28. The command `A>chkdsk b:*.* /v` will
a. show all the files
b. show only fragmented files
c. show no file names
d. cause an error message

_____ 29. If you use `CHKDSK` without any switch, DOS will actually fix a corrupted file and store recovered pieces in files named `FILEnnnn.CHK`. True or false?

_____ 30. Before a hard disk can be formatted, it must be partitioned with `FDISK` (or a comparable program). True or false?

Summary and Review

Check each item as you review and understand the following highlights of the chapter.

☐ 1. The most commonly used storage media are the 3.5-inch and 5.25-inch floppy disks. While differing in size, they share common arrangements: write-protect notch (switch), indexing device, read-write opening, and protective cover.

☐ 2. A floppy disk contains a fragile (floppy) piece of plastic coated with metal oxide. It should be treated with great care and protected from many harmful elements. If not, it may not reliably store data or stored data may be lost.

☐ 3. DOS uses its `FORMAT` command to prepare a disk for use. Formatting divides a disk into a number of concentric rings called tracks, each of which is divided into a number of subunits called sectors, each of which has 512-byte storage capacity.

☐ 4. The `FORMAT` command has many available switches. The most important is the `/S` switch. It transfers three system files (two hidden files plus `COMMAND.COM`) to the disk being formatted. This will make the new disk bootable. which means that you can start DOS with this disk.

☐ 5. If no `FORMAT` switch is used, a disk is formatted to its full capacity. Thus, a 1.2MB drive formats a 5.25-inch quad-density disk to 1.2MB capacity.

☐ 6. As more types of disks become available, compatibility has become a serious problem. For example, 3.5-inch and 5.25-inch drives/disks are never compatible. A quad-density drive (1.2MB or 1.44MB) can read and write to a double-density disk (360KB or 720KB); the reverse is not true.

☐ 7. You may use different drives in different places, home and office, for example. In that case, you need to use different switches of `FORMAT` to make your disks more compatible. If you want a disk to be usable in both drives, you need to format to the lower capacity of the two.

☐ 8. If you use version 4, the `/F` switch allows you to format a disk to a specified number of bytes. For earlier versions, you need to use `/T`, `/N`, or other switches to format to a lower capacity so the disk can be used in another drive.

☐ 9. Besides tracks and sectors, formatting also creates a boot record, two FATs, and a directory. This overhead reduces each disk's capacity for storing data.

☐ 10. When DOS stores data on a disk, it takes up at least one cluster for a file. A cluster may be 1 to 16 sectors, depending on the type of disk. Thus, a 1-byte file could eat up disk space of 512 bytes to 8KB.

☐ 11. The internal `VOL` command can be used to display a disk's volume label, which can be created when `FORMAT` is used with the /V switch.

☐ 12. The external `LABEL` command can be used to show, create, change, and delete a volume label.

☐ 13. Use the external command `CHKDSK` to show a variety of useful information, including RAM and disk space. Use the `/F` switch to fix corrupted files and the `/V` switch to display file names.

☐ 14. A hard disk needs to be partitioned with `FDISK` and then formatted with the `/S` switch to be bootable. A large-capacity hard disk can be divided into a number of drives, one of them made bootable and the rest used just for data storage.

Review Questions

Write an answer for each question below.

1. Explain CD-ROM and DAT.
2. Compare and contrast the different types of floppy disks in use today.
3. Explain the compatibility problems of 360KB and 1.2MB drives and disks (720KB and 1.44MB if you use 3.5-inch floppy disks).
4. Explain the different external parts of a floppy disk.
5. How does `FORMAT` divide a disk?

6. What items are created by FORMAT to manage a disk?
7. What is a cluster? What is its significance?
8. Explain the purposes of the /S and /B switches of FORMAT.
9. What does FORMAT's /F switch do?
10. Explain the purposes of VOL and LABEL.
11. What items are displayed on the screen by using CHKDSK without any switch?
12. What do CHKDSK's /F and /V switches do?
13. How can you find out whether the files on a disk are contiguous or fragmented?
14. What does the RECOVER command do?
15. How is a hard disk organized?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
FORMAT	Cluster size not shown after formatting	Cluster size shown after formatting
FORMAT /F	Switch not available	New switch
FORMAT /V	/V required if you want to be prompted to enter a volume label; otherwise, no prompt	If switch is not used, you'll be prompted to enter a label. You can also enter a label like this: FORMAT /V:diskname
CHKDSK	Allocation units not shown	Shows allocation units
FDISK	Can create a hard disk partition up to 32MB	Can create a hard disk partition up to 2GB

Creating and Naming Files

Chapter 4

TOPICS:

Creating Files with `COPY CON`
Appending to a File
DOS File Name Conventions
Wild Card Characters
DOS Editing Keys
Creating and Using Batch Files
DOS Commands and Batch Files

COMMANDS COVERED:

`COPY` (`COPY CON`), `DIR`, `TYPE`, `DEL`, `REN`

NEW TERMS AND SYMBOLS:

DOS device name, reserved word, `CON`, `AUX`, `COM1`, `PRN`, `NUL`, `LPT1`, wild card characters, `?`, `*`, DOS editing keys, template, batch file, replaceable parameter (dummy variable), `%`, `AUTOEXEC.BAT`

REQUIRED MATERIALS:

A practice disk.

Quick Start

✎ **To create a file:**

```
A>copy con filename
```

✎ **To append from the keyboard to an existing file:**

```
A>copy filename + con
```

✎ **To avoid illegal characters in file names:**
See the list on page 97.

✎ **For a list of DOS device names:**
See Table 4.1.

✎ **To use the wild card characters ? and *:**
See the Wild Card Characters section.

✎ **To copy matching files:**

```
A>copy file? b:
```

✎ **To delete all the files:**

```
A>del *.*
```

✎ **To rename matching files:**

```
A>ren file?.txt *.doc
```

✎ **To use DOS editing keys:**
See Table 4.2.

✎ **To create and use a batch file:**
See the Creating and Using Batch Files section.

So far you have learned to use various DOS commands such as COPY, DIR, and TYPE to handle existing files. From now on, you will learn how to create your own files. Then you will use that knowledge to create batch files that DOS can execute. If you use batch files often, you can save a great deal of time and vastly increase your productivity.

While creating files, you need to observe DOS file name conventions and avoid illegal names. This chapter teaches you which file names to use and which to avoid. You can take advantage of the DOS editing keys to avoid repetitive typing and thus simplify your work. After studying this chapter, you will know how to recycle previously entered commands.

When you handle existing files, you can utilize wild card characters (also known as global file name characters) to speed up access to the files that match certain specifications. In this chapter you will learn how and where to use these characters.

Creating Files with COPY CON

- **Use COPY CON to create a file.**
- **End file creation by entering the ^Z character.**

To create a short file, you can use the COPY command combined with CON and a file name. When you are done, DOS will copy from what you have entered from the keyboard (and displayed on the screen) to the named file on disk. Follow these steps to create a file:

1. Place a practice disk in drive A and enter the following:

```
A>copy con chap1.txt
```

2. Type the following lines and press Enter at the end of each:

```
This is line one.
This is line two.
```

3. Press Ctrl-Z or F6 to display the ^Z character.

   The ^Z (ASCII 26) character, which is treated by DOS as a single character, is the end-of-file marker. When DOS encounters this character, it considers a text file to have ended.

4. Press Enter to save the file.

   After the above steps, the screen appears as follows:

```
A>copy con chap1.txt
This is line one.
This is line two.
^Z
        1 File(s) copied
```

DOS tells you that one file has been copied. This means that the keystrokes entered from the keyboard and displayed on the screen have been copied to the disk file named CHAP1.TXT.

5. Display the file's contents:

```
A>type chap1.txt
```

The two lines in the file should be displayed on the screen.

Tips & Tricks

File Creation and Existing Files

When creating a file, beware of potential conflict with an existing file or directory. If you use the name of a file already in existence in the current directory, the new file will replace the old. You will not be warned that an existing file is to be replaced. If you are not careful, you could inadvertently wipe out an important file.

If you use the name of a subdirectory of the current directory, you will not succeed in creating a file. Suppose you have created a directory named `CHAP1.TXT` as a subdirectory of the root directory. If you are in the root directory and try to use `COPY CON` to create a file named `CHAP1.TXT`, DOS will redisplay your typed lines on the screen and tell you that one file has been copied. It is copied to the screen, not to disk. The existence of an identical directory name will prevent DOS from creating a file of the same name. Directories are explained in the next chapter.

Appending to a File

✎ **Append lines to an existing file by using this command:**
`COPY FILENAME + CON`

Sometimes you want to append (add) lines to an existing file. You could use `COPY CON` to create a file with the same name and retype everything, plus the new lines. Or you could just add the new lines to those already in existence. To do the latter, simply follow these steps:

```
A>copy chap1.txt + con
CHAP1.TXT                                   (added by DOS)
CON                                         (added by DOS)
This is line three.                         (typed by you)
^Z
        1 File(s) copied
```

The `COPY CHAP1.TXT + CON` command (space before and after the + is optional) tells DOS to add from the keyboard (console) to the named file already in existence. When you press Enter, DOS responds by displaying the first two lines in

uppercase, telling you that the second (CON) will be appended to the first (CHAP1.TXT).

If the file does not already exist, the file name is not displayed and only CON is shown. Whatever you enter, after you enter ^Z, DOS will display the text on the screen but not send it to the nonexisting file. Here is an example:

```
A>copy chap2.txt + con
CON                                    (added by DOS)
This line is not saved.                (typed by you)
^Z
This line is not saved.                (displayed by DOS)
        1 File(s) copied
```

If you reverse CON and the name of an existing file, the file contents will be added to the screen, as illustrated below:

```
A>copy con + chap1.txt
CON                                    (added by DOS)
This line won't be saved.              (typed by you)
^Z                                     (lines below displayed by DOS)
This line won't be saved.
CHAP1.TXT
This is line one.
This is line two.
This is line three.
        1 File(s) copied
```

Here DOS interprets the command as adding to the screen the contents of the existing file. It displays CON, but not the file name, at the beginning. After you type lines and enter ^Z, the new lines are repeated, plus those from the named file.

Using COPY CON to create or append to a file is, as shown above, very simple. However, you cannot edit a file with it. After Enter is pressed, a line is no longer available for modification. If a file is more than a few lines long or needs frequent changes, a better tool is EDLIN, the line editor that is part of the DOS package. EDLIN will be covered in Chapter 10. After you learn it, you will use it to create and edit longer batch files in Chapters 11 and 12.

Tips & Tricks

The Syntax of COPY CON

You may be puzzled by this syntax of COPY CON:

```
A>copy con filename
```

This syntax does not seem so perplexing if you remember this syntax of COPY:

```
A>copy source target
```

When DOS encounters a command like this:

```
A>copy con filename
```

it interprets your intent as copying from CON to FILENAME. Here CON is the console (keyboard and monitor combined). What comes out of the console is copied to disk; a disk file is thus created. Conversely, you can copy from a disk file to the console, such as:

```
A>copy filename con
```

This will display the named file on the screen. Its use will be more fully explained in Chapter 6.

Practice 4-1

- ☐ 1. Use COPY CON to create a file named QUOTE containing this line:
  Live simply so others may simply live.
- ☐ 2. Add this line to QUOTE:
  Learn to read so you may read to learn.
- ☐ 3. Rename QUOTE as QUOTES.
- ☐ 4. Display the contents of QUOTES on the screen.

DOS File Name Conventions

- ✎ **Certain characters are not allowed as part of a file name.**
- ✎ **A DOS device name cannot be used as a file name.**

You have been asked to enter file names on many occasions. Mostly, however, you have been asked to enter the names of existing files. That posed no problem; all you

needed to do was to enter the exact names. In this chapter, you are asked to enter new file names. It poses a new problem for you because you cannot use just any names. Before we proceed further, you need to know what file names are permitted.

DOS imposes a few major constraints on a file name:

- **It can be no more than eight characters long, plus a maximum of three characters placed after a period for an optional extension.**
- **Every file name in the same directory must be unique, i.e., different from another.**
- **Certain characters are not permitted.**
- **DOS device names cannot be used as file names.**

Maximum Length

A file name can be no longer than eight characters, plus an optional extension of no more than three characters; the name and the extension are separated by a period. While a file name can be as short as a single character, it cannot be longer than this maximum:

```
FILENAME.EXT
```

If you enter more than eight characters without inserting a period, those beyond the eighth character are ignored and discarded. If you have more than three characters after the period, only three are retained.

Illegal Characters

In creating a file, you cannot use as part of a file name any of the characters listed below:

```
(space) ? * " / \ [ ] : | < > + = ; ,
```

These characters are used by DOS for various purposes. Most of them are shown and briefly explained in Appendix C (see the `SYMBOLS` item).[1] They will be more fully discussed in later chapters.

Also excluded are the ASCII characters below 32, in addition to 127.[2] These characters, obtained by holding down Ctrl and pressing a letter key (or holding down Alt and typing a number on the numeric keypad), are used to control computer hardware and communications. See Chapters 10–12 for examples of their uses.

If you use any character in either of the above categories, you will get the `Invalid number of parameters` or `File creation error` message.

[1] You can also use `DOS.EXE` to see how these characters are used by DOS. Just enter this command:

```
A>dos symbols
```

[2] Entering ASCII 127 has the same effect as pressing the Backspace key.

DOS Device Names

DOS device names are **reserved words** that have special meanings to DOS and should not be used as your file names. These are shown in Table 4.1.

Table 4.1 DOS Device Names

Device Name	Purpose
CON	Console (keyboard and screen)
AUX	Auxiliary device (serial port)
COM1/2/3/4	Communications (serial) ports
PRN	Printer (parallel port)
LPT1/2/3	Line printers (parallel ports)
NUL	Nonexistent (dummy) device

If you use any of these as a file name, DOS will interpret your intent as using one of these devices, not creating a file.

Unlike the special characters, a device name can be part of a file name. For example, while CON alone cannot be a file name, it can be combined with other legal characters to form a legal name. Thus, CONSOLE, AUXCON, and so on are perfectly legal names.

Special Characters

Except those mentioned above, all characters and combinations are permitted, including numbers and the extended ASCII characters not shown on the keyboard. To enter a character not shown on the keyboard, hold down the Alt key, type a number on the numeric keypad (not the top row of the keyboard), and release the Alt key. For example, if you want to use the Japanese yen sign as a file name, you can follow these steps:

1. Type the following:

   ```
   A>copy con _
   ```

2. Hold down Alt, type 157 on the numeric pad, and release the Alt key to display this:

   ```
   A>copy con ¥
   ```

3. Press Enter and type text.
4. Press F6 and Enter to end.

If you now use DIR, you will find a newly created file named ¥. If you enter that character (hold down Alt and type 157) after the TYPE command, you can display the contents of that file. (The ASCII characters are listed in Appendix B.)

Extensions to Avoid

While it is possible to use file name extensions of COM and EXE, it is not advisable. These are executable files created by compilers or assemblers. A file created with COPY CON cannot be executable just because you add such an extension. Besides, you could unwittingly replace a program that comes with DOS. Most damaging of all, if you try to run such a file (one created with COPY CON carrying COM or EXE), it will most likely lock up your computer, requiring rebooting.

If you use programs that follow special extensions in naming the files they create, such as BAS for BASIC, WK1 for Lotus 1-2-3, and PLN for PlanPerfect, you should also avoid using these extensions in the files you create. These files carry special formatting codes recognizable by the programs that create them.

Avoiding Wild Cards

The wild card characters of ? and * should also be avoided while creating a file name. Characters placed after * are completely ignored and discarded; if they are placed at the beginning of a file name, such as COPY CON *ABC, DOS will not create any file.

The ? character produces peculiar results. Placed in the first three positions, it can make CON part of your file name. Placed after the third position, it is simply ignored. Two examples are shown below:

```
COPY CON ???123?.DOC          CON123.DOC
COPY CON B???VET.?XT          BONVET.XT
```

The wild card characters, however, are most useful when you want to do something with existing files, as explained in the next section.

Tips & Tricks

A Blank in a File Name

You cannot use a space (pressing the space bar, or ASCII 32) in a file name. You can, however, use the blank character of ASCII 255. Either character shows nothing on the screen, but DOS treats them differently. You can, for example, create a file like this:

```
A>copy con no way
There is no way a novice can peek into
this file.
^Z
```

After typing `NO`, hold down Alt and type 255 on the numeric keypad. Then type `WAY`. If you just press the space bar where the gap is shown, you will get this error message instead:

```
A>copy con no way
Too many parameters
```

When you want to use `DEL`, `DIR`, `REN`, `COPY`, `TYPE` and other commands on this file, you need to supply ASCII 255 where the gap is.[3]

Drill

_____ 1. If there is an existing file named `FILENAME` located in the current directory and you enter the command `A>copy con filename`, DOS will NOT create a new file. True or false?

_____ 2. The command `A>copy con con` will create a file named `CON`. True or false?

[3] If your keyboard does not do anything after you press Alt-255, you need to take an extra step. Place both `KEYB.COM` and `KEYBOARD.SYS` files in drive A (the root directory). If you use version 4, enter this command:
`A>keyb us`
If you use version 3.3, enter this instead:
`A>keyb us,437,keyboard.sys`

_____ 3. Which file name below is illegal?
   a. A-B
   b. A_B
   c. A+B
   d. A^B

_____ 4. Which file name below is legal?
   a. US..$$$
   b. BR/#
   c. FRANK.!!!
   d. JAN(!).YEN

Wild Card Characters

- ✎ **Use ? to match one single character in a particular position of a file name.**
- ✎ **Use * to match all the characters from a particular position.**

The **wild card characters**, question mark (?) and asterisk (*), can be used on existing file names to speed up your work. The ? is used to represent any single character; the * applies to multiple characters. They are useful because DOS will automatically find matching files when you want to do something about them. This will save you the trouble of repeatedly typing specific file names that are similar.

The ? Character

Follow these steps to demonstrate the use of the ? wild card character:

1. Create a new file:

```
A>copy con chap2.txt
This is from chapter 2.
^Z
```

2. Show matching files:

```
A>dir chap?.txt
 . . .
CHAP1      TXT
CHAP2      TXT
 . . .
```

If you have not erased the file previously created, two files should show up.

When DOS encounters the ? character, it will substitute any character found in a specific position. For example, if you enter this command:

```
A>dir chap?.txt
```

DOS will display the following files, if they exist:

```
CHAP.TXT
CHAP1.TXT
CHAP2.TXT
CHAPA.TXT
CHAPB.TXT
. . .
```

In short, any file name with the pattern of CHAP, followed by zero or one character, followed by the TXT extension, will be considered matching.

If you want to copy all the related files, you can simplify typing by entering a simplified command like this:

```
A>copy chap?.txt b:
```

This will copy all the matching files shown above from drive A to drive B, saving you the trouble of copying each file separately.

If you have a file named CHAP12.TXT, what can you do? The single ? will not match more than one character. You can, however, use the character twice or more:

```
A>copy chap??.txt
```

This command will copy the following:

```
CHAP.TXT
CHAP1.TXT
CHAP12.TXT
```

In short, any file matching the fixed characters, plus zero or up to two of the missing characters, will be acted on. If there are three characters missing, such as CHAP123.TXT, the file will not be identified.

When the ? character is not placed at the end of a file name or extension, every character and position must match. Suppose you have these files:

```
CHAPTXT
CHAP1TXT
CHAP2TXT
```

The command below will list the last two, but not the first:

```
A>dir chap?txt
```

The * Character

If you use the * character, it will match anything missing. For example, if you enter this command:

```
A>dir chap*.*
```

DOS will display these files, if they exist:

```
CHAP
CHAP1.TXT
CHAP123.ABC
CHAP1234.ANY
```

In other words, any file starting with CHAP, plus zero or up to four more characters, plus zero or up to three extension characters, will all be considered matching.

When DIR is used, DOS will loosely match many files. If you omit the extension, DOS will assume that you want to include it. This command:

```
A>dir chap*
```

will do the same things as the one in the preceding example; it is treated the same as CHAP*.*. On the other hand, if you put a period at the end, like below:

```
A>dir chap*.
```

you ask DOS to show only those matching files without extensions. Those with extensions will not be considered matching. If you enter a command like this:

```
A>dir *.
```

DOS will display all the files without extensions. However, this command:

```
A>dir .*
```

will display everything, just like DIR *.*, or simply DIR alone. The DIR command's default setting is global, meaning everything is included.

The above examples will not necessarily apply to the DEL (or ERASE), REN, or COPY commands. These will require an exact match before an action is taken. For example, this command:

```
A>copy chap*
```

will copy only CHAP, CHAP1, CHAPXXXX, and so on, i.e., any file starting with CHAP, plus zero or up to four more characters. However, it will not copy any file with an extension, such as CHAP1.TXT. In short, it is treated the same as with a period added at the end. Thus the following will copy and rename only the matching files without extensions:

```
A>copy chap* work*
```

If you want DOS to act on all the existing files in a directory, use the global specifier of *.* with a specific command (except DIR as explained earlier), such as:

```
A>copy *.* b:
```

This command will copy everything from drive A to drive B.

Wild card characters can be applied to REN as well. You can thus quickly rename a lot of files this way:

```
A>ren chap*.txt *.doc
```

All the matching files with the TXT extension will keep their original names but with the new DOC extension. Using * in the target can also save typing time when working on a single file, such as:

```
A>ren longname.txt *.bak
```

This can save retyping a long file name. The above is thus a shortcut of below:

```
A>ren longname.txt longname.bak
```

The Danger of Using ? and * with DEL

Using a wild card character, particularly *, with DEL or ERASE is dangerous and you should use it with great caution. Consider the following:

```
A>del chap*
```

It will delete as many files as it copies, as shown above. There are, however, two important reasons for you to be careful. First, it does not warn you that multiple files will be erased. Second, unintended files could be erased. Before you enter such a

sweeping command with potentially disastrous consequences, you should make sure that you know what you are doing.

The only time DOS gives you a warning is when you want to delete everything in a directory, such as:

```
A>del *.*
All files in directory will be deleted!
Are you sure (Y/N)?_
```

This command generates the warning message and seeks confirmation from you. If you enter Y, everything in the current directory of the default drive will be erased. An identical warning and (Y/N) prompt will also appear when you want to delete the contents of an entire subdirectory, such as:

```
A>del misc
```

If MISC is a file name in the current directory, no warning is given. If it is a subdirectory name, however, DOS interprets your order to mean erase all the files in that subdirectory. If you enter Y, every file there will be erased.

Version Variances: Realizing the potential danger of combining DEL with a wild card, Microsoft added the /P (Prompt) switch to version 4. If you have it, you should take advantage of it, such as:

```
A>del chap* /p
```

When a matching file is found and targeted for deletion, its name and path will be shown and you will be asked to enter Y or N. Only if you enter Y will DOS delete a displayed file.

Tips & Tricks

The Period as the Current Directory

In Chapter 5 you will learn that a period (.) alone as a parameter is interpreted by DOS as the current directory. You can use it to replace *.* where it is appropriate, such as:

```
A>copy . b:
A>del .
A>echo y | del .
```

The space before the period is optional; DOS can recognize your intention even without this space. The first command will copy every file in the current directory in drive A to the current directory in drive B. The second command will delete everything; you will be prompted with Y/N. The third will automatically supply the Y without waiting for your response. Do NOT use this destructive command unless you know what you are doing. If you want to experiment with it, create a separate directory and some nonsense files. The ECHO command is discussed in Chapters 11 and 12.

TYPE and Wild Cards

You cannot use a wild card when a command can act on only one file at a time. For example, if you try this:

```
A>type chap?.txt
Invalid filename or file not found
```

DOS will refuse to oblige, and give you an error message instead. You cannot use TYPE to display the contents of more than one file at a time.

One way to bypass this limitation is to use this command instead:

```
A>copy chap?.txt con
```

This will display the contents of all the matching files one after another.

Another way is to use TYPE with the FOR...IN...DO command. This command will be covered in Chapters 8 and 11.

Practice 4-2

- ☐ 1. Create a file named QUOTA and put a line of text in it.
- ☐ 2. Use ? characters with DIR to show both QUOTES and QUOTA.
- ☐ 3. Use DEL and * to delete both files.

Drill

_____ 5. The command A>del day??.* will delete all of the following, except:
   a. DAY
   b. DAYON.TRA
   c. DAYNA.PAY
   d. DAYSON.WK1

_____ 6. The command A>dir file*. will show all but:
   a. FILE123
   b. FILE12.3
   c. FILE1234
   d. FILES123

_____ 7. When you enter the DEL *.* command, DOS will ask you to enter Y/N before deleting everything in the current directory. True or false?

_____ 8. The command `A>del .` will:
   a. delete all the files in drive A's current directory
   b. give a warning and a `Y/N` prompt
   c. both a and b
   d. cause an error message due to an illegal command

_____ 9. The command `B>copy . a:` will copy
   a. all the files from drive B to drive A
   b. some files from B to A
   c. all the files from A to B
   d. nowhere to nowhere; it's illegal

_____ 10. The `DIR` command alone works the same way as `DIR *.*`. True or false?

_____ 11. The command `A>dir chap*.` will match a file named `CHAP22.TXT`. True or false?

_____ 12. The command `A>type *.*` is legal. True or false?

DOS Editing Keys

✎ **Use the first five function keys (F1–F5), plus the Esc, Ins, Del, and Backspace keys to save time when entering DOS commands.**

When you enter a DOS command or use `COPY CON` to create a file, you can take advantage of the **DOS editing keys** to simplify your work. Skillful and selective use of these keys can save you time in typing DOS commands.

The Template and Editing Keys

When you type a command and press Enter, this command is saved in a memory buffer called a **template**. When you enter a new command, it replaces the old one in the template. What is stored in the template can be copied to the screen in order to avoid repetitive typing.

Suppose you have most recently entered the following:

```
A>type chap1.txt
```

Now you want to redo the same thing. You can retype the whole line and press Enter to display the text. Or you can just press F3. The whole line reappears on the command line. If you then press Enter, you can quickly reissue that command.

Table 4.2 summarizes what you can do with these editing keys. It shows that with F3 you can copy from the template and display on the screen everything from the cursor position to the end of the template. If the cursor is at the beginning of the command line when F3 is pressed, the entire previous line is copied. If not, the portion before it is skipped.

Table 4.2 DOS Editing Keys

Key	Function
F1 (or Right Arrow)	Copy one character from template to screen
F2c	Copy all characters before first typed c
F3	Copy all characters from cursor to end of line
F4c	Skip characters before first typed c (then use F1 or F3 to copy the rest)
F5	Store the current line in the template
F6	Insert ^Z character
Esc	Abandon edited line; start anew
Ins	Insert mode on/off
Del	Skip over a character in template
Backspace (or Left Arrow)	Delete to left a displayed character

(c denotes a character)

Think of a template as an exact mirror image of the line you have most recently entered. If you want to recycle this line, you can use the keys in Table 4.2 to do it. In addition to using the F1–F4 keys to copy the whole or portions of that line to the screen, you can insert extras at proper places after pressing the Ins key once, or skip over certain characters in the template by pressing Del an appropriate number of times. The Ins and Del keys are trickier than the rest because you do not immediately see what is done to the template.

Steps in Using Editing Keys

To practice using the DOS editing keys, let us follow these steps:

1. Put a practice disk in drive A and create a file by entering the following:

```
A>copy con chap1.txt
This is line one.
This is line two.
^Z
```

2. Display the contents of the file:

```
A>type chap1.txt
```

3. Type `DIR` and press F3:

```
A>dire chap1.txt
```

Notice that `dir` has replaced `typ`, but the `e` still remains. Pressing F3 at this point copies from the template to the screen the remainder of the line, i.e., from the fourth

character to the end. If you press Enter at this point, DOS will give you an error message because `DIRE` is not a legal command.

4. Press Esc to abandon the line:

```
A>dire chap1.txt\

_
```

A backslash appears at the end of the line and the cursor moves to the beginning of the next line. (Another alternative is to press Backspace repeatedly to erase everything and then start anew from the beginning of the same line.) You can now start again. Any time you want to abandon an edited line, just press Esc. Nothing is changed, and the abandoned line does not replace the one still in the template.

5. Type `DIR`, press Del once, press F3, and press Enter:

```
A>dire chap1.txt\
 dir chap1.txt
```

By pressing Del after typing `DIR`, one character in the template is skipped over and the cursor is positioned at the next character; you do not see the result right away, but DOS is keeping track. Pressing F3 at this point copies only the remaining characters, those from the cursor position to the end of the line.

6. Type `TYPE` and press F3:

```
A>typechap1.txt
```

If you press Enter now, you will get an error message. You need to insert a space between the command and the file name for DOS to recognize your intention.

7. Press Esc to abandon the edited line:

```
A>typechap1.txt\
```

8. Press F1 until `TYPE` appears, press Ins once, press the space bar, press F3, and press Enter:

```
A>typechap1.txt\
  type chap1.txt
```

Press Ins once to activate Insert mode, twice to turn it off. When Insert is on, characters you type are inserted at the cursor location and the rest are pushed rightward. If Insert mode is not on, characters you type replace existing ones.

You cannot tell from the screen or the keyboard whether the Insert mode is on or off. You basically have to feel it in the dark. Pressing Enter or another editing key returns you to the default off mode. To turn Insert on again, press Ins once more.

9. Press F4, press the space bar once, press Ins once, type `DIR`, press F3 and Enter:

```
A>dir chap1.txt
```

When you press F4 followed by the space bar, the characters before the first space are skipped over. Pressing F3 after that will copy only the characters located from it to the end of the line. By pressing Ins once and typing `DIR` before pressing F3, you insert `DIR` and copy the original characters not skipped over.

10. Press F2, type c, type *, press Del four times, press F3 and Enter:

```
A>dir *.txt
```

By pressing F2 followed by c, all the characters before the first c (lowercase) are displayed on the screen. Pressing Del each time skips over one character in the template. The screen does not show the skipping until you press F3 to copy the unskipped portion.

11. Type *.TXT and press F5:

```
A>*.txt@
_
```

After pressing F5, the @ sign appears to mark the end of the line and the cursor moves down to the beginning of the next line. DOS does not execute this line; it puts the line in the template instead, replacing the previously entered line.

12. Press Ins once, type DIR, press the space bar once, and press F3:

```
A>*.txt@
  dir *.txt
```

The *.txt string previously stored in the template is now copied to the screen after you press F3.

So far we have demonstrated the use of all the keys in Table 4.2 except F1 and Backspace. F1 is used to copy from the template and display on the screen one character at a time. You can also use the Right Arrow key to do the same thing. The Backspace key is used to erase to the left a character displayed on the screen. The cursor moves leftward as each character is erased. You can also use the Left Arrow key for the same purpose.

Using Editing Keys with COPY CON

DOS editing keys can be used with COPY CON to duplicate the whole or portions of a previous line. Let us use the file you created in step 1 to demonstrate how this can be done. First, enter the following:

```
A>copy con chap1.txt
This is line one.
_
```

After you press Enter and the cursor moves down to the next line, the previous line is put in the template. If you want to duplicate part of this line in the new line, you can use a proper editing key to avoid retyping the whole line. One way to do this is to press F3 to copy the previous line, then press Backspace or the Left Arrow key four times to erase "one.", then type "two." and press Enter.

Another way is to press F2, then press o (lowercase) to copy everything before this character, then type "two." and press Enter.

Instead of saving this file by entering ^Z, let's break it. Hold down Ctrl, press C, and release both. The ^C character (treated by DOS as a single character) appears at the end:

```
A>copy con chap1.txt
This is line one.
This is line two.
^C

A>_
```

Your effort to copy from the keyboard to a file on disk is terminated. No file is created. If there is an existing file with the same name, it remains unaltered.

Ctrl-C is often used to terminate a DOS function or a computer program in progress; you can also use it to terminate a batch file or program being executed. If it does not work, try the Ctrl-Break combination. Some programs intentionally disable either combination to prevent premature termination; they will usually provide other ways for an orderly exit.

A Mnemonic System

The function keys used for editing seem quite confusing. After using them for a long time, I have developed a mnemonic system to associate each key with a specific function:

F1	One at a time
F2	To (Two)—up to—a typed character
F3	Three's a crowd, whole bunch
F4	From (same first letter as Four) a typed character
F5	File (rhymes with Five) to template

This system may help you remember what to do. If not, you may have a better idea of your own.

Practice 4-3

☐ 1. Use `COPY CON` to enter this line:
`You can't be all things to all people.`

☐ 2. Use F2, DEL, and F3 to add a second line:
`You can be all things to all people.`

☐ 3. Use F4, Ins, and F3 to add a third line:
`Be all things to all people.`

☐ 4. Abort (cancel) the file creation.

Drill

Find an answer in the list below to match questions 13–24:

a. F1, b. F2, c. F3, d. F4, e. F5, f. F6, g. Esc, h. Del, i. Backspace, j. Ins.

_____ 13. Copy all remaining characters from the template.

_____ 14. To add to what is already in the template.

_____ 15. To skip over a character in the template.

_____ 16. Abandon an edited line.

_____ 17. Store a line in the template without execution.

_____ 18. Copy before the first typed character.

_____ 19. Skip before the first typed character.

_____ 20. To delete to the left a displayed character.

_____ 21. Can be substituted with Left Arrow.

_____ 22. Copy one character.

_____ 23. Works the same way as Right Arrow.

_____ 24. To end a file creation.

Creating and Using Batch Files

Now that you have learned to create files, you can create batch files. A **batch file** contains a batch (bunch) of commands, which DOS will execute one after another. It is a very useful tool that can save you time and labor as you work with your computer.

Batch File Requirements

A batch file has to meet certain conditions. It must

- **contain only ASCII text**
- **include commands DOS understands**
- **have a `BAT` extension in its file name**

First, it must be ASCII text that DOS can read. What you create with `COPY CON` or `EDLIN` is ASCII text. A file you create with your word processor may not be ASCII. In that case, DOS may not recognize or follow your commands.

Second, a batch file must contain commands DOS understands. If not, error messages may be generated or intended tasks may not be performed. The DOS commands we have covered so far can all be entered into a batch file.

Third, a batch file must have the file name extension `BAT`. If this extension is missing, DOS will not execute the commands in it. Do not use `EXE` or `COM` as an extension for a DOS batch file. If you try to execute such a file, you will most likely lock up your computer.

In naming a batch file, you should avoid the name of an internal or external DOS command. This can avoid confusion between DOS commands and your batch files. We will return to this topic after you know how to create and use batch files.

Batch File Examples

Let us follow these steps to create a batch file:

1. Put a practice disk in drive A and enter the following:

```
A>copy con da.bat
```

This command tells DOS to create a file named DA.BAT. The file name is arbitrary, but the BAT extension is mandatory.

2. Type this command and press Enter:

```
dir a: /p
```

This command instructs DOS to list the files in drive A one screen at a time.

3. Press F6 (or Ctrl-Z) and Enter to save the file. The screen now shows the following:

```
A>copy con da.bat
dir a: /p
^Z
        1 File(s) copied
```

4. Enter the following command to display the contents of the newly created file:

```
A>type da.bat
```

5. Use the new batch file by entering this:

```
A>da
```

After step 5, the directory information of drive A should be displayed on the screen. With the existence of this batch file, you can just enter DA (instead of DIR A: /P) to show what is in drive A.

Suppose you now repeat the above steps to create two more batch files with the names and contents shown below:

```
DB.BAT  DIR B: /P
DC.BAT  DIR C: /P
```

From now on, any time you want to know what is in a particular drive, just enter DA, DB, or DC.

This arrangement requires you to maintain three separate files. Couldn't we combine them into one? Yes. To do that you need to understand the concept of a **replaceable parameter** (also known as a **dummy variable**), which is designated as % followed by a number (0–9). Any time DOS finds this sign in a batch file, it will replace it with a parameter you enter on the command line.

Follow these keystrokes to create a batch file:

```
A>copy con d.bat
dir %1: /p
^Z
```

This file differs from the others by one character. Instead of a drive letter, you find the % sign followed by 1 (parameter number 1).

When you use this batch file, you need to enter the name of the file, plus the expected parameter, a drive letter to replace %1. For example, if you want to show drive C, all you need to do is to enter the following:

```
A>d c
```

The D is for D.BAT (you can enter the entire file name if you so wish) and C is for drive C. This tells DOS to run the D.BAT batch file and put C in place of %1. DOS will treat the single command in the batch file as:

```
dir c: /p
```

If you want to show another directory, just enter a different parameter. Instead of C, you can enter A, B, or any other drive you may have.

The batch file name (D) and the parameter (C) must be separated. If you put them together, as in DC, DOS will execute DC.BAT instead. If the file does not exist, an error message will follow.

If no parameter follows the batch file name, DOS will fail to carry out your wish. Without a parameter, the command inside the batch file becomes:

```
dir : /p
```

In this case, DOS is asked to list a file named ":". Nothing will be listed because the colon character cannot be used as a file name.

If the D.BAT file is in drive A and the default drive is changed to another one, such as B, you need to provide a path so that DOS knows where to find the batch file to execute, such as:

```
B>a:d b
```

This command tells DOS to go to drive A to run D.BAT and to display the files in drive B.

One variation of this batch file is to take out the colon and keep the rest, like this:

```
A>copy con d.bat
dir %1 /p
^Z
```

In this case, if you specify no parameter when running the batch file, DOS will use the default drive as the target instead of giving you an error message. On the other hand, if you want to target a different drive, you must include a colon like this:

```
A>d b:
```

Without the colon, DOS will interpret your command as showing a file named B.

AUTOEXEC Batch File

✎ **An AUTOEXEC.BAT file on the boot disk is automatically executed after DOS is booted.**

An **AUTOEXEC.BAT** file is an automatic executable batch file. Every time DOS is booted, it looks for such a file, if it exists, to execute.[4] Such a file allows you to instruct DOS to automatically execute many commands without any human intervention. A lot of time can be saved and a lot of work can be done automatically if you can use this device skillfully.

An AUTOEXEC.BAT file must meet certain conditions before DOS will execute it automatically:

[4] You can also run such a batch file by entering the name on the command line.

- **The name must be exact. Thus you can have only one such file on a boot disk.**
- **The file must be stored in the root directory of the boot disk, together with the COMMAND.COM file.**
- **The file must contain commands DOS can execute.**

Creating an AUTOEXEC.BAT file follows the same steps as in the creation of any other file. Let us now create such a file with the following commands:

```
A>copy con autoexec.bat
date
time
path=c:\bat
^Z
```

An AUTOEXEC.BAT file will be created and stored on the disk in drive A. When this file is executed, DOS will first show this:

```
A>date
Current date is Sun  6-24-1990
Enter new date (mm-dd-yy): _
```

This is the familiar scene explained in Chapter 1. If the date is correct, you can just press Enter. If not, enter a new date. Then the next command is executed:

```
A>time
Current time is 10:15:31.58
Enter new time: _
```

Give it the same treatment as the date item.

If the DATE and TIME commands are not present in an AUTOEXEC.BAT file, DOS will not pause to let you enter new information. Instead, the computer's internal time is used. If your computer has a battery-powered clock, you should take out both the DATE and TIME commands. If not, they are necessary for you to enter accurate numbers.

The PATH=C:\BAT command is executed after the DATE and TIME commands are taken care of. If you have a hard disk and consistently save all your batch files to a directory named C:\BAT, this command allows you to run a batch file regardless of where the default drive is. With this arrangement, you no longer need to worry whether DOS can find a batch file to execute. The PATH command will be explained fully in Chapter 5.

Hard Disk Users: Most batch file examples in this section use a hard disk to store batch files. This is the most logical arrangement. Other ways, however, will also work. If you do not use a hard disk, you can use A or B instead of C in your batch files.

If you have a hard disk and work with several software packages, you can take advantage of an AUTOEXEC.BAT file to speed up your access to a particular software. One way to handle this situation is to create a number of batch files with similar names, such as AUTOEXEC.WP for WordPerfect, AUTOEXEC.123 for Lotus 1-2-3, and so on. These files are not executable because of the absence of a BAT extension. However, when the time comes for you to work continuously with a package, all you need to do is to change the extension to BAT.

To create a batch file for WordPerfect, enter the following:

```
C>copy con autoexec.wp
cd\wp51
wp
cd\
^Z
```

The batch file for Lotus 1-2-3 may look like this:

```
C>copy con autoexec.123
cd\lotus
123
cd\
^Z
```

When the time comes to work with WordPerfect, just use the REN command to change AUTOEXEC.WP to AUTOEXEC.BAT. That way, every time the computer is turned on, the current directory will be changed and WordPerfect will be loaded. When you exit WordPerfect, the current directory will be changed to the root directory again.

The CD command is to change the current directory. Like PATH, it will be explained in the next chapter.

This arrangement is suitable if you work with an application continuously over a period of time. If you alternate among different packages, each for a short period of time, the following arrangement may be better:

```
C>copy con w.bat
cd\wp51
wp
cd\
^Z

C>copy con l.bat
cd\lotus
123
cd\
^Z
```

This way, after DOS is booted, you can just enter L to run Lotus 1-2-3 or W to start WordPerfect.

If you have quite a few packages you use alternatively, you may consider building a menu batch file to show you available options and let you press a single key to start a program. This device will be explored after you learn more batch commands in Chapter 11.

There are many uses for batch files. This chapter has given you just a hint of what is to come. Where appropriate, future chapters will exploit batch files to save you time and increase your productivity.

Batch files can be structured like computer programs. You can use certain commands to control program flow. These commands are explained in Chapter 11. They will be used to write more sophisticated batch files in Chapters 11 and 12.

Speedy Creation of Batch Files

If you create batch files often, you may want to employ a batch file to speed up your work. This batch file will do the trick:

```
A>copy con b.bat
copy con %1.bat
^Z
```

From now on, every time you want to create a batch file, just enter B, followed by a file name, minus the BAT extension. Suppose you want to create an AUTOEXEC.BAT file. You can just enter the following:

```
A>b autoexec
```

The screen will be ready for your work by showing the following:

```
A>b autoexec
copy con autoexec.bat

_
```

After you enter appropriate commands and end with ^Z, a batch file will be created. This saves you the trouble of typing a long command at the beginning.

If you forget to enter a file name, the screen will show this instead:

```
A>b
copy con.bat

_
```

Since you do not have a legal file name, you should not continue. Press Ctrl-C. You will be asked whether you want to terminate the batch job. Enter Y and rerun the B.BAT file to start anew.

If you want to automatically execute a batch file right after you finish creating it, this arrangement will do the trick:

```
A>copy con b.bat
copy con %1.bat
%1
^Z
```

Suppose you want to test a batch file again and again. You can simply run the batch file this way:

```
A>b test
copy con test.bat
. . .

^Z
test
```

The first line is automatically displayed and you can start entering your batch lines. After you enter ^Z at the end, the last line automatically executes the batch file (TEST.BAT) that has just been created and saved.

DOS Commands and Batch Files

Earlier you were warned to avoid using a DOS command as the name of a batch file. If you use an internal command as a batch file name, you can never access this file. Suppose you create this file:

```
A>copy con dir.bat
dir b: /p
^Z
```

This will result in `DIR.BAT` being saved to drive A. However, you can never use this file for its intended purpose. If you issue `DIR`, the internal command will be executed, preventing you from reaching the batch file saved on disk.

Aha, you say, I can foil DOS by using this command:

```
A>dir.bat
```

No use. DOS will interpret your command to mean this:

```
A>dir *.bat
```

That is, showing the files matching a particular extension. Your batch file remains untouched.

An external command's name can be used as a batch file name. However, you need careful planning to avoid confusion. For example, if you have `CHKDSK.COM` and `CHKDSK.BAT` stored on the same disk, the latter can never be executed. If you issue this command:

```
A>chkdsk
```

DOS will first try to execute `CHKDSK.COM`. If `CHKDSK.COM` is not accessible, `CHKDSK.EXE` is next. Only then will `CHKDSK.BAT` take its turn.

As long as a batch file and a matching external DOS command are not stored in the same directory, the batch file can be executed without any problem. Suppose you store all your external DOS commands in the DOS directory in drive C and nowhere else. You can create a batch file and save it to drive A, like this:

```
A>copy con chkdsk.bat
c:\dos\chkdsk %1
^Z
```

You can run the batch file without a parameter, like this:

```
A>chkdsk
```

This command will execute `CHKDSK.BAT` from drive A. The batch file will then go to the DOS directory in drive C to run the `CHKDSK.COM` program and check the disk in drive A. Since no parameter is specified, the `%1` is ignored and the default drive (A) is the target.

You can run the batch file with various parameters, like below:

```
A>chkdsk /v
A>chkdsk filename
A>chkdsk b:*.*
```

The first will check the disk and list all the files in drive A. The second will check the contiguity of `FILENAME` in drive A. And the third will check the contiguity of all the files in drive B. These parameters were explained in Chapter 3.

Practice 4-4

- ☐ 1. Create a batch file named `F.BAT` to reduce the keystrokes normally needed to format a disk. Assume that `FORMAT.COM` is stored in `C:\DOS` and the target drive is A. (If you have a two-floppy system, use A as the source and B as the target.)
- ☐ 2. Alter the above batch file and change the fixed target drive of A to a replaceable parameter. Use the batch file to format a disk in drive B.
- ☐ 3. Use `B.BAT` discussed in the text to create `CK.BAT`, which will load `CHKDSK.COM` from drive A and check a disk specified from the keyboard as a parameter.
- ☐ 4. Create an `AUTOEXEC.BAT` file that will automatically run a computer program, such as BASIC, as soon as DOS is booted.

Drill

_____ 25. Ctrl-Z is used to cancel (abort) batch execution. True or false?

_____ 26. An `AUTOEXEC.BAT` file can be placed in any directory and DOS will execute it. True or false?

_____ 27. Assuming all the files are in the current directory, the command `A>prog` will execute which file?
a. `PROG.COM`
b. `PROG.EXE`
c. `PROG.BAT`
d. `PROG.DOC`

_____ 28. `MYBAT.BAH` is a legal batch file name. True or false?

_____ 29. As soon as DOS is booted, it will automatically try to execute a file named `AUTO.BAT`. True or false?

_____ 30. You can never execute a batch file named `COPY.BAT`. True or false?

Summary and Review

Check each item as you review and understand the following highlights of the chapter.

- ☐ 1. Use `COPY CON` followed by a file name to create a file. Type the lines to be saved to the file. Press F6 and Enter to end.
- ☐ 2. When creating a file, do not use the name of an existing file in the current directory, unless you mean to replace the old file with the new one. There is no warning that a file is to be replaced.
- ☐ 3. Do not use the name of a subdirectory as the name of a file to be created. If you do, DOS will not save the file you intend to create, but echo the file to the screen instead.
- ☐ 4. Observe DOS file name conventions when creating or renaming a file. A file name can be no longer than eight characters, plus an optional extension of up to three characters.
- ☐ 5. Do not use certain characters, such as ¦ and > . These are used by DOS for special purposes such as piping and redirection.
- ☐ 6. Do not use DOS device names as file names; also avoid DOS commands under most circumstances.
- ☐ 7. Do not use the wild card characters ? and * in a file name when using `COPY CON` to create a file.
- ☐ 8. Use wild card characters when acting on existing files. Each ? character represents a character of the same position in a file name. Each * represents all the remaining characters up to the end of a file name or extension.
- ☐ 9. Use the global specifier of * . * (can be replaced with a single period) to speed up actions on all the existing files in a specific directory, such as copying and deleting. Use it with caution when combined with `DEL` or `ERASE`. If you use version 4.0 or later, use the `/P` switch often as a precautionary measure.
- ☐ 10. Use `COPY CON` to create short batch files. Use batch files to save time and increase productivity. Use a batch file to speed up creation of other batch files.
- ☐ 11. For DOS to be able to execute it, a batch file must have the extension `BAT` in the name and contain only ASCII text and commands DOS can execute.
- ☐ 12. Create an `AUTOEXEC.BAT` file and store it on the boot disk for DOS to automatically perform certain initial tasks so you do not have to do them manually.

Review Questions

Write an answer for each question below.

1. If you enter the `COPY CON FILENAME` command, what are you asking DOS to do?
2. When you use `COPY CON` to create a file, how do you end?
3. If you decide not to create a file after using `COPY CON`, how do you abort it?
4. Suppose you want to add lines to an existing file named `WORK`. What command do you use?
5. Name five characters that cannot be used in a file name.
6. Can you use characters not on the keyboard as parts of a file name? How?
7. Can you use the wild card characters `?` and `*` in naming a file?
8. Can you use wild card characters when deleting files?
9. Can you use wild card characters with `TYPE` or `RENAME`?
10. What will this command do?

```
A>copy file* con
```

11. What happens if you press Esc after typing a DOS command?
12. If a batch file is stored in the `C:\BAT` directory, what can you do to run the batch file from any drive or directory?
13. What's the purpose of using a replaceable parameter in a batch file?
14. Where do you store an `AUTOEXEC.BAT` file so that DOS can execute it as soon as it is booted?
15. What will happen if you name a batch file with a `COM` or `EXE` extension and then execute it?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
`DEL /P`	Switch not available	New switch

Managing Directories

Chapter 5

TOPICS:

Organizing a Disk
Making and Removing Directories
Maneuvering Related Directories
DEL (ERASE) and Directories
Maneuvering Multilevel Directories
Maneuvering Unrelated Directories
Maneuvering the Prompt
Using TREE to Display Directories
Using PATH to Access Executable Files
Accessing Data Files with APPEND

COMMANDS COVERED:

MD (MKDIR), RD (RMDIR), CD (CHDIR), TREE, PROMPT, PATH, APPEND

NEW TERMS AND SYMBOLS:

Directory, subdirectory, parent directory, root directory, current directory, \, ., . ., metastring, $, TSR, DOS environment

REQUIRED MATERIAL:

1. A practice disk
2. Access to TREE.COM and APPEND.EXE

Quick Start

✎ **For samples of organizing a hard disk:**
See Figures 5.1 and 5.2.

✎ **To create a subdirectory:**
```
A>md practice
```

✎ **To remove a subdirectory:**
```
A>rd practice
```

✎ **To change to a subdirectory:**
```
A>cd practice
```

✎ **To change to the root directory:**
```
A>cd\
```

✎ **To change to one level above the current directory:**
```
A>cd..
```

✎ **To display a directory path:**
```
A>cd
```

✎ **To use metastrings with `PROMPT`:**
See Table 5.1.

✎ **To create a DOS prompt showing drive, directory, and >:**
```
A>prompt $p$g
```

✎ **To display a treelike structure of directories:**
```
A>tree b:
```

✎ **To be able to run batch files stored in the `C:\BAT` directory:**
```
A>path c:\bat
```

✎ **To access data files in both `C:\BAT\TXT` and `B:\FILES`:**
```
A>append c:\bat\txt;b:\files
```

✎ **To show a `PATH` or `APPEND` list:**
```
A>path
A>append
```

Storage media continue to be built with larger and larger capacity. Today's hard disks commonly have up to several hundred megabytes (million bytes) of storage capacity. CD-ROMs can go up to gigabytes (billion bytes). Even floppy disks continue to expand in storage capacity while their physical size continues to shrink. A few years ago, 5.25-inch 360KB disks were standard. Today 5.25-inch 1.2MB and 3.5-inch 1.4MB floppies are common. The next wave may be 3.5-inch 4MB floppies.

How do you handle a large-capacity disk? You divide it into smaller units and subunits, and store related information in a particular unit. Then you use an index or catalogue through which you can quickly find what you need. This is comparable to a library putting different kinds of books in different areas and using a catalogue to locate a specific book. Many things in human society are organized this way: telephone directories, a book's table of contents, and so on.

DOS follows the same idea in handling your disk. It allows you to divide a disk into many directories (units) and subdirectories (subunits). It also provides quite a few tools for you to manage these directories. These directories and the tools to manage them are discussed in this chapter.

Organizing a Disk

- **A directory structure can be created to suit your specific needs.**
- **Using directories to store files has many advantages.**

To visualize a disk and its directory structure, think of a file cabinet. A file cabinet may have several drawers, which can be compared to a disk's directories. A drawer may be divided into several sections, which are comparable to subdirectories. In addition, each directory also serves as an index, keeping track of files stored in it; this is analogous to your putting labels in proper places to keep track of things.

The concept of a directory should not be new to you. When DOS formats a disk, as explained in Chapter 3, it creates a root directory. When you save a file to the disk, the file name, extension, size, date, etc. are saved to the directory. When you use the DIR command, DOS uses the saved data to show you the files (and other related data) in a directory. When you create a subdirectory, as explained below, DOS also manages it in the same way it does the root directory.

Basic Terms

There are a few commonly used terms you need to be familiar with to better understand the concept of **directories.** In the beginning, there is the **root directory.** Every disk has a root directory, which is created during formatting. If you do nothing to divide up your disk, it has only one directory, the root directory.

If you divide a disk into units and subunits, you have directories and subdirectories. These are terms used to relate one directory to another. The active directory is the **current directory**; the root directory is always the current directory if a disk is not divided. The one above the current directory (the next higher level),

if any, is the **parent directory**. The one below it (the next lower level), if any, is a **subdirectory** (some people refer to it as a child directory). As you go closer to the root directory, you go to higher levels. As you go further to the subdivisions, you go to lower levels.[1]

Tips & Tricks

A Hard Disk's Root Directory

When a colleague of mine first got a PC installed in his office, he copied all the files from many floppy disks to the hard disk without using any subdirectories. When he used the DIR command on the hard disk, a long, long list of files scrolled across the screen. It was a mess.

If you have a large-capacity hard disk, 40MB or more, you normally should not store any file in the root directory except these:

```
COMMAND.COM
CONFIG.SYS
AUTOEXEC.BAT
```

These, plus the hidden system files, must be in the root directory for DOS to boot from the hard disk.

The rest of the root directory should store only subdirectories. As you acquire a new software package, create a subdirectory and install the package in it. This arrangement keeps the root directory relatively uncluttered and allows you to quickly find what you need.

Sample Organizations

Directories and subdirectories can be organized in many ways. In a system used by several persons, the hard disk may be like the structure shown in Figure 5.1. In this arrangement, each person occupies a directory, which is a subdirectory of the root directory. In each person's directory, subdirectories can be created to store related files. This way, when Jim wants to find his correspondence, he can go to the directory bearing his name and from there to his `LETTR` subdirectory.

[1] A lower level directory goes into the branches, and a higher level directory goes closer to the root. This is comparable to dangling a tree upside down, with the root at the top.

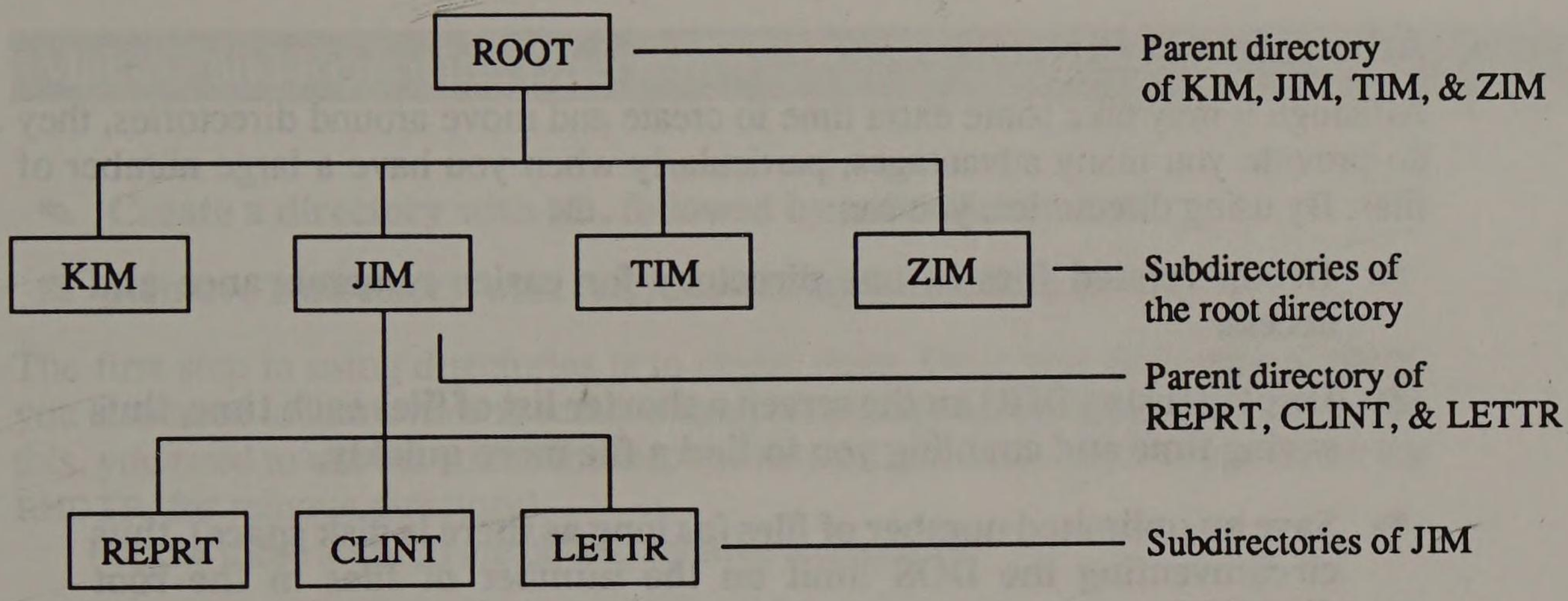


Figure 5.1 A multiuser directory structure

If you use a hard disk all by yourself, you may want to organize it according to functions, tasks, or software packages. Figure 5.2 could serve as a model.

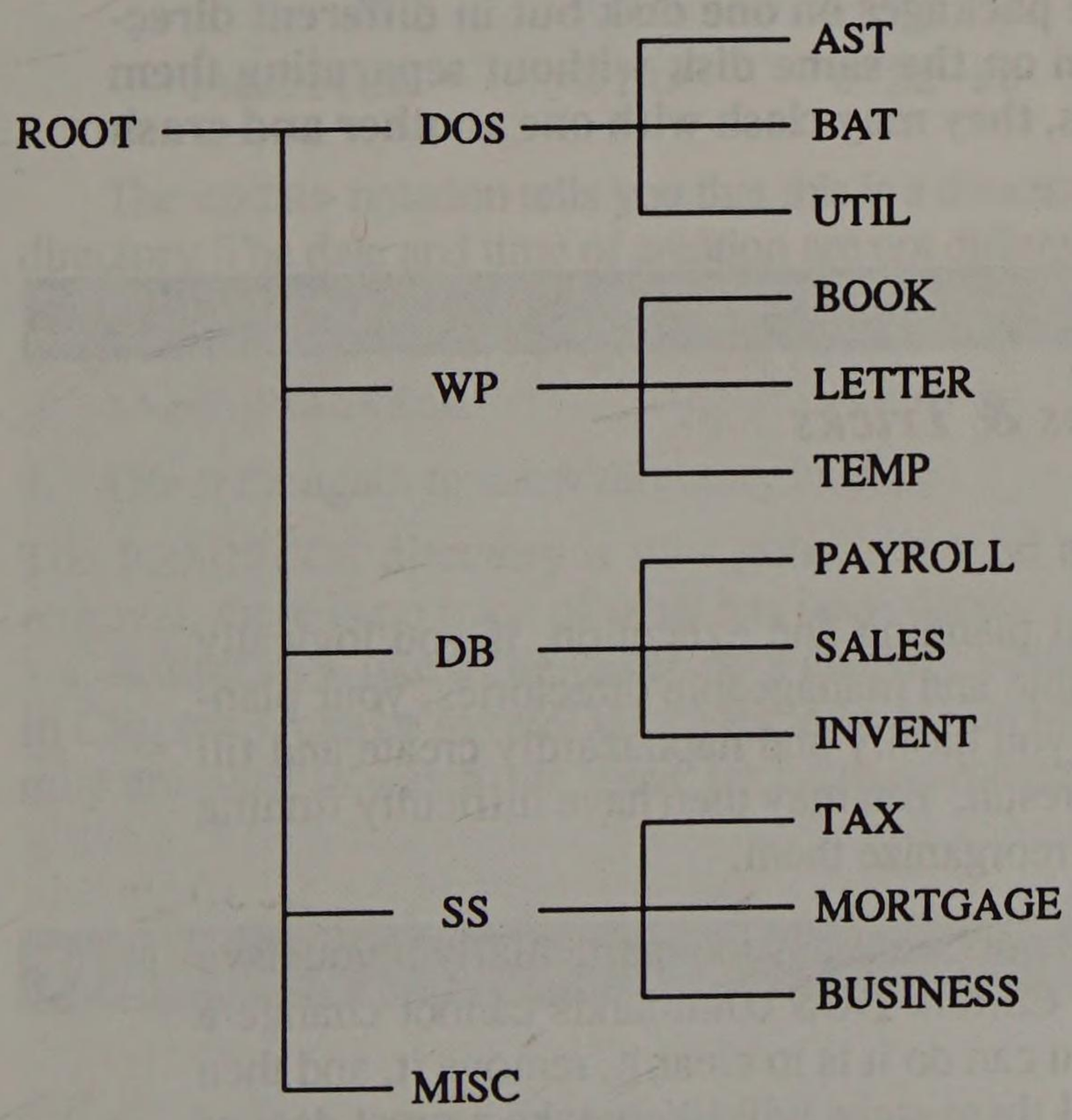


Figure 5.2 A single-user directory structure

In the above example, instead of lumping all the files in one directory, you divide them into five directories. In this arrangement, you can go to the WP directory to do word processing, to the SS directory to work on spreadsheets, and so on.

Advantages of Directories

Although it may take some extra time to create and move around directories, they do provide you many advantages, particularly when you have a large number of files. By using directories, you can:

- **Group related files in one directory for easier remembrance and access.**
- **Display (using DIR) on the screen a shorter list of files each time, thus saving time and enabling you to find a file more quickly.**
- **Save an unlimited number of files (as long as there is disk space), thus circumventing the DOS limit on the number of files in the root directory.**
- **Use identical file names that are stored in different directories.**
- **Share a large-capacity disk with other users, perhaps coworkers or fellow students, with a relatively well-defined territory for each.**
- **Store different software packages on one disk but in different directories. If you store them on the same disk without separating them into different directories, they may clash with one another and crash your computer.**

Tips & Tricks

Careful Planning for Directories

Using directories requires careful planning and execution. If you logically group your files into understandable and manageable directories, your planning will pay off handsomely. If you hastily and haphazardly create and fill directories, confusion will likely result. You may then have difficulty finding your files, or spend more time to reorganize them.

Reorganizing your directories will not be an easy job, particularly if you have numerous multilevel directories. Current DOS commands cannot change a directory name. The only way you can do it is to clear it, remove it, and then recreate it with another name. All these steps will likely take a great deal of time, which could be avoided if you spend some time planning your directory strategy.

Even with careful planning, you can sometimes get confused about where things are and how to find them. When this happens, you can enlist the help of a few DOS tools. These include the `PROMPT`, `TREE`, `PATH`, and `APPEND` commands, all of which are explained in this chapter.

Making and Removing Directories

- ✎ **Create a directory with MD, followed by a directory name.**
- ✎ **Remove a directory with RD, followed by a directory name.**

The first step in using directories is to create them. Once you have created them, you also need to know how to remove them when they are no longer needed. To do this, you need to use the commands MD (or MKDIR, for make directory) and RD (or RMDIR, for remove directory).

Follow these steps to create and remove a directory:

1. Put a practice disk in drive A and enter this command:

   ```
   A>md practice
   ```

2. Display directory entries to see what you have done:[2]

   ```
   A>dir
   ```

The screen shows your old files, if any, plus a directory you just created:

```
. . .
PRACTICE      <DIR>      6-22-90    7:22p
. . .
```

The <DIR> notation tells you that this is a directory or subdirectory of the current directory. The date and time of creation are not different from regular DOS files.

3. Remove the newly created directory:

   ```
   A>rd practice
   ```

4. Use DIR again to show directory entries.

The PRACTICE directory is now gone, removed by the RD command. After the removal, there is no trace of what has been done.

A directory, like a file, takes up a cluster of disk space; a cluster, as explained in Chapter 3, can be several kilobytes long. If you have a small number of files, you may not want to waste this space for extra directories.

Practice 5-1

- ☐ 1. Insert a practice disk in drive A and create a subdirectory named TEST.1.
- ☐ 2. Prove the existence of TEST.1 by showing it on screen.
- ☐ 3. Remove TEST.1 and prove its removal.

[2] You can also show the subdirectory entries by putting the subdirectory name ofter DIR, like this:

```
A>dir practice
```

Tips & Tricks

Directory Names

DOS treats a directory name as a file name. In entering a directory name, you need to follow DOS file name conventions. You can have no more than eight legal characters, plus an optional extension of up to three characters. These conventions were explained in Chapter 4.

You are not allowed to use the name of a file in the current directory as a directory name. Suppose you have a file named `PRACTICE` located in the root directory. DOS will not let you create a subdirectory named `PRACTICE` from the root directory. Having two identical names, even though one is for a file and the other for a directory, will create confusion and is thus illegal.

Drill

_____ 1. Before you can create a directory, you must first use `MD` to create a root directory. True or false?

_____ 2. It is NOT a good idea to put two large software packages, such as WordPerfect and Lotus 1-2-3, in one directory. True or false?

_____ 3. Suppose there is already a file named `MISC` in the current directory. The command `A>md misc` will create a directory named `MISC`. True or false?

_____ 4. The `REN` command can be used to change a directory name. True or false?

Maneuvering Related Directories

- **Use `CD` (or `CHDIR`) to change to a different directory.**
- **Use \ to designate the root directory.**

After you create a directory, you need to know how to get in or out of that directory or how to make use of it. The following steps illustrate some simple uses:

1. Create a test file:
```
A>copy con testfile
This is a test file.
^Z
```
2. Create a directory named `BACKUP`:
```
A>md backup
```

3. Go to the new directory:

```
A>cd backup
```

The CD (or CHDIR, for change directory) command is used to change to the directory specified after the command. The above command tells DOS to go to the named directory, which is a subdirectory of the current directory.

4. Display the files in the directory:

```
A>dir
```

The screen shows the following:

```
 Volume in drive A has no label ———————— Disk volume label
 Directory of  A:\BACKUP ———————————————— Directory path

.              <DIR>     6-22-90   7:46p —— Current Directory
..             <DIR>     6-22-90   7:46p —— Parent Directory
        2 File(s)     198656 bytes free
```

The first line shows the volume label name for this disk. The second line indicates the location of the current directory and how it is related to the root directory. The A:\BACKUP path shows you that the BACKUP directory is a subdirectory of the root directory.

What you see here is an empty directory with no file but two directory notations. The single period (.) indicates the current directory and the two periods (..) represent the parent directory. While there is no file stored in the current directory, DOS tells you there are two files, one for the current directory and the other for the parent directory. In counting the number of files, DOS treats a directory as a file.

5. Copy a file from the root directory to the current directory:

```
A>copy \testfile
```

The backslash (\) is used to indicate the root directory. The command above tells DOS to copy the named file from the root directory to the unspecified target directory (here meaning the current directory). If you want the laborious way, you can try either command below:

```
A>copy \testfile \backup
A>copy \testfile \backup\testfile
```

The first command tells DOS to copy a file named TESTFILE, which is in the root directory, to the directory named BACKUP, which is one level below the root directory; the name of the file is not changed. The second adds a file name to the target directory. If you want a name different from the original, you can change it here.[3]

You can also copy a file from another drive to drive A's current directory, like this:

```
A>copy b:filename
```

This command will copy FILENAME from drive B's current directory to drive A's current directory.

[3] Although these extra entries are optional here, they may be necessary when the target directory is different from the current one.

6. Display the directory entries again:

```
A>dir
```

This time you see the name of a new file you just copied into this directory, displayed below two directory notations.

7. Go to the root directory:

```
A>cd\
```

Putting a backslash (\) after the CD command always takes you to the root directory no matter where you are.

8. Try to remove the directory:

```
A>rd backup
```

What do you get? Does the following message surprise you?

```
Invalid path, not directory,
or directory not empty
```

The reason for this message is that the directory you want to remove is not empty.

9. Delete the file in the subdirectory:

```
A>del \backup\testfile
```

This command tells DOS to go to the BACKUP directory to delete the named file.

10. Try again to remove the directory:

```
A>rd backup
```

This time, DOS willingly obliges. The directory is gone, after the file stored there has been erased.

Tips & Tricks

Removing a Directory

You cannot remove the current directory or another directory directly above it. That is illegal. If you could do that, where would you be after the current directory or another one above it is removed?

Furthermore, you cannot remove a directory that has anything stored in it. When DOS detects any contents in a directory, it will refuse an order to remove it. It is a wise precaution. Without this safeguard, you could inadvertently remove a directory together with a group of files.

Some copy-protected software packages can create some nasty side effects on your hard disk. After you clear a directory by deleting all the files, something invisible is still left there. Even though the directory is empty, you cannot remove it with currently available DOS commands. If this happens to you, you may have to follow the manual's prescribed way to remove it.

DEL (ERASE) and Directories

✎ **DEL or ERASE can be used to erase only files, not directories.**

You cannot use the DEL or ERASE command to remove a directory; you can use it only to delete files. Suppose you are in the root directory and enter this command:

```
A>del backup
```

If BACKUP is a file, it is immediately erased. If it is a directory, you will be asked:

```
All files in directory will be deleted!
Are you sure (Y/N)?_
```

DOS assumes that you want to erase every file stored in the BACKUP directory and asks you to confirm. If you enter Y, the files are erased but the directory remains. So, if you intend to clear a subdirectory and then remove it, enter the following commands:

```
A>del backup
All files in directory will be deleted!
Are you sure? (Y/N)y
A>rd backup
```

The above two commands produce the same results as the following four commands:

```
A>cd backup
A>del *.*
A>cd\
A>rd backup
```

Warning: Using DEL to clear an entire subdirectory could be hazardous. You may not know what is stored in the subdirectory. If you want to be sure, go to the subdirectory and check what is in it first.

Version Variances: If you have version 4, you can use the new /P (Prompt) switch with DEL, such as:

```
A>del backup /p
```

Each file targeted for deletion, together with its path, will be displayed and you will be asked for Y or N. If you enter Y, the file is deleted and the next matching file is shown.

Practice 5-2

- ☐ 1. Insert a practice disk in drive A and create a directory named TEMP.
- ☐ 2. Make TEMP the current directory.
- ☐ 3. Copy a file from drive B to the TEMP directory.

- [ ] 4. Go to the root directory of drive A and delete all the files in the TEMP directory.
- [ ] 5. Remove the TEMP directory.

Maneuvering Multilevel Directories

✎ **In a multilevel directory structure, you need to provide a full path if you want to move to a desired directory. Use backslashes (\) to separate directory names.**

You can use a number of ways to create and navigate in a multilevel directory structure. The following steps show you how to create multilevel directories and move from one to another:

1. Create a first-level directory:

```
A>md\dir1
```

The above command creates a directory named DIR1. The \ is optional if you are in the root directory; it could be substituted with a space, as illustrated previously.

Tips & Tricks

Space versus Backslash

Inserting a space and backslash after MD (CD or RD) have different implications. With a space, DOS assumes you want to create a subdirectory of the current directory. If you put \, DOS assumes you want to create a subdirectory of the root directory. The two happen to have the same effect when you are in the root directory. If you are not, they cause different consequences.

2. Create a second-level directory:

```
A>md\dir1\dir2
```

This tells DOS to create DIR2, which is a subdirectory of DIR1, which in turn is a subdirectory of the root directory. DIR1 must exist before DIR2 can be created. If not, an error message appears. Instead of the above command, you can use the CD command to go to the DIR1 directory and then create a subdirectory from there. Thus, the above command could be replaced by the following two commands:

```
A>cd dir1
A>md dir2
```

In the second command above, you cannot use \. If you did, DOS would assume that you intended to create a first-level directory (a subdirectory of the root directory), not a subdirectory of the current directory.

3. Create a third-level directory:

```
A>md\dir1\dir2\dir3
```

Again, you are creating a directory that is three levels below the root directory. All the backslashes are necessary to tell DOS where each fits in. If you go to the level 2 directory, you can create a subdirectory from there as illustrated in the above alternative method.

4. Go from the root to the level 2 directory:

```
A>cd\dir1\dir2
```

Since you are in the root directory to begin with, you need to use backslashes to guide DOS to a specific place. In this case you tell DOS to go to `DIR2`, which is a subdirectory of `DIR1`, which in turn is a subdirectory of the root directory. The above command could be replaced with the following:

```
A>cd dir1
A>cd dir2
```

In this alternative method, you tell DOS to go to the next (lower) level one step at a time.

5. Display the files:

```
A>dir

 Volume in drive A has no label
 Directory of  A:\DIR1\DIR2

.            <DIR>          3-22-90   8:20p
..           <DIR>          3-22-90   8:20p
DIR3         <DIR>          3-22-90   8:20p
        3 File(s)     196608 bytes free
```

Although there is no file stored in this directory, the screen shows three files in existence. As mentioned earlier, DOS counts a directory as a file. The above screen tells you that there is a subdirectory named `DIR3`. That plus the current directory (designated with a single period) and the parent (marked with two periods) make up the three files. If you go to the `DIR3` directory, you will see only two files listed—one for the current and one for the parent.

6. Display the root directory's entries:

```
A>dir\
```

Even though you are in another directory, you can always use \ to tell DOS to do something on the root directory. This command does not change the current directory; you are still in `DIR2`.

7. Display level 3 directory entries:

```
A>dir dir3
```

This command tells DOS to display the directory that is a subdirectory of the current directory. Do not include \ here. If you do, you tell DOS that DIR3 is a subdirectory of the root directory. This will lead to an error message in our case.

8. Go to the root directory:

```
A>cd\
```

9. Display the level 3 directory entries:

```
A>dir\dir1\dir2\dir3
```

This command displays the files in DIR3, but it does not change the current directory. The root directory remains the current directory.

Maneuvering Unrelated Directories

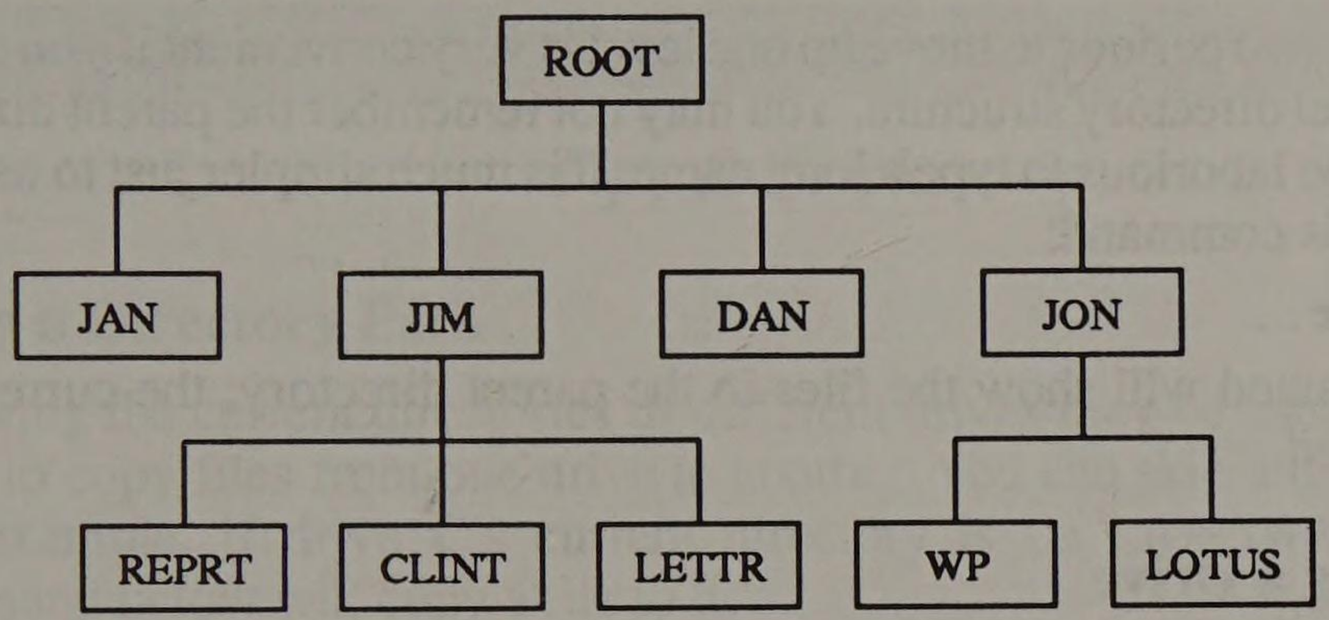
- **To move between two directories not directly related, you need to provide a specific path.**
- **Use CD followed by two periods (. .) to move to the parent directory.**
- **CD can change a directory but not a drive. To change a drive, enter a drive letter followed by a colon (e.g., C:).**
- **Use CD alone to display the directory path of a drive. Knowing the directory path of the source or target drive can save time when copying, displaying, or deleting files.**

Moving from one directory to another directory, if the two are not directly related, requires special manipulation. One way to visualize the problem is to treat directories as branches of a tree. DOS rules do not allow you to jump like a monkey from one branch to another. You must go to where the branches meet and from there to a new branch. To travel from one directory to another unrelated directory, you need to direct DOS to the connecting point and branch from there to the new directory.

In the directory structure below, you may be tempted to jump directly from REPT to CLINT without going through the connecting point (JIM). If you did that, however, you would get the error message "Invalid directory."

Changing to Unrelated Directories

To illustrate how you can navigate among a complex directory structure, let us suppose that you have in drive C a structure like this:



The list below provides some examples of going from one directory to another, not directly related, directory:

To go from this directory	To this directory	Enter this command
REPRT	LOTUS	C>CD\JON\LOTUS
LETTR	DAN	C>CD\DAN
JAN	WP	C>CD\JON\WP
DAN	REPRT	C>CD\JIM\REPRT
LOTUS	JAN	C>CD\JAN
JON	REPRT	C>CD\JIM\REPRT
WP	CLINT	C>CD\JIM\CLINT
REPRT	CLINT	C>CD..\CLINT

Let us examine the first and last examples. The first example shows how to get from REPRT to LOTUS. The command tells DOS to go to the root directory (the first \), branch from there to JON, and then branch from there to LOTUS. This command could be replaced with the two below:

```
C>cd\
C>cd jon\lotus (or cd\jon\lotus)
```

The second command above can also be replaced with these two:

```
C>cd jon
C>cd lotus
```

Two Periods as the Parent Directory

The last command in the table uses two periods (. .) to signify the parent directory. This command tells DOS to go up one level and then branch from there to the CLINT subdirectory. This command could be replaced with the following two:

```
C>cd..
C>cd clint
```

You should not use \ in the second command shown above. If you did, DOS would consider CLINT to be one level below the root directory. This would generate an error message in our example.

Using two periods to move up one level is very convenient if you are deep inside a multilevel directory structure. You may not remember the parent directory's name or it may be laborious to type a long name. It is much simpler just to use two periods, such as this command:

```
C>dir..
```

This command will show the files in the parent directory; the current directory is not changed.

Changing a Drive

Although CD can change the current directory, you cannot use it to change the default drive. For example, if you enter the command in the first line, the second line shows that the default drive is still A:

```
A>cd c:\
A>_
```

What you have done here is to change the current directory in drive C to the root directory. You are still in drive A. To go to drive C, you need to change it with this command:

```
A>c:
```

After the above two commands, you are now in the root directory of drive C.

Showing the Current Directory

You can use the CD command alone to make DOS show you the current directory. The screen may display something like this:[4]

```
A>cd
A:\DOS
```

You can also specify a drive to show its current directory:

```
A>cd c:
```

This may show a result like below:

```
C:\JON\WP
```

[4] Notice there is no > sign. This is what distinguishes it from the display produced by PROMPT, explained in another section below.

Tips & Tricks

Skipping a Directory Path

Knowing the current directories of different drives may be useful when you want to copy files from one drive to another. You can skip a directory path. For example, if drive C's current directory is `C:\JON\WP`, the simple command below will copy a file to it:

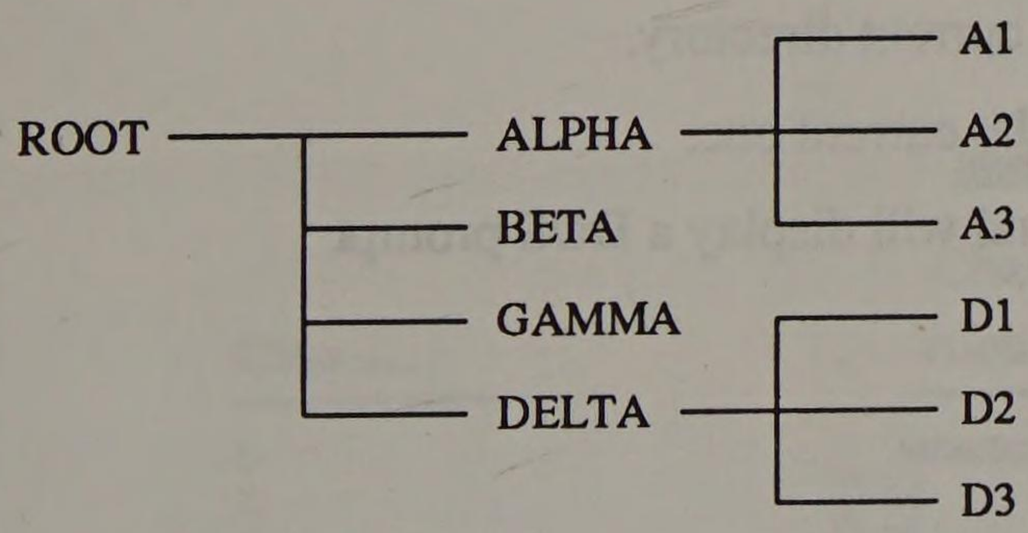
```
A>copy filename c:
```

If drive C's current directory is not known or not the target directory you have in mind, you will need to specify it, such as:

```
A>copy filename c:\jon\wp
```

This command will copy `FILENAME` from the current directory of drive A to the `C:\JON\WP` directory.

Practice 5-3



- ☐ 1. Create on a practice disk the directory structure shown above:
- ☐ 2. Make `A1` the current directory and copy a file to it from drive `A`'s root directory.
- ☐ 3. Keep `A1` as the current directory, but copy a file from there to `D3`.
- ☐ 4. Change the current directory to `D1`.
- ☐ 5. Display the entries in the root directory.

(Save the directory structure for the next session.)

Drill

_____ 5. If you enter the RD TEMP command, you will:
   a. remove the TEMP directory located anywhere
   b. remove the current directory if named TEMP
   c. always remove a subdirectory named TEMP
   d. not necessarily do anything

_____ 6. If you enter the DEL TEMP command where TEMP is a subdirectory of the current directory, you will:
   a. erase the directory's name
   b. erase the files in that directory
   c. cause an error message
   d. remove the directory

Select from the list of commands below to answer questions 7–11:

   a. CD\
   b. CD
   c. CD.
   d. CD..
   e. CD...

_____ 7. Make the root directory the current directory.

_____ 8. Not a valid command.

_____ 9. Show the current directory and default drive.

_____ 10. Do nothing; keep the same current directory.

_____ 11. Make the parent directory the current one.

_____ 12. Used alone, the CD command will display a DOS prompt like this:
   a. A:\>
   b. A:\
   c. B:\DOS>
   d. C>

Maneuvering the PROMPT

- **Use the internal PROMPT command and a metastring character to display default drive, current directory, date-time, etc.**
- **Use PROMPT alone to restore the default DOS prompt.**

One tool you can use to locate where you are in a maze of complex directories is the PROMPT command. You can use it to display the current directory as the DOS prompt. Once you know where you are, you can tell where other things are.

Metastring Characters

PROMPT is followed with a text string or a **metastring**, which consists of $ and another character. A metastring can transform a character to another character or an action as explained below.

When used with a regular text string, PROMPT simply displays that text string as a prompt. You could, for example, enter this command:

```
A>prompt Hello:
```

Here you just type the PROMPT command, space, and then the text string. The space can be replaced with an equal sign (=) without making any difference (this also applies to PATH and APPEND discussed below), such as

```
A>prompt=Hello:
```

After the command, the new prompt becomes:

```
Hello:_
```

If you want to return to the original (default) prompt, simply enter the PROMPT command without any string attached. The familiar A> prompt reappears. Or, if you want to be more elaborate, you can use this:

```
prompt $n$g
```

The $n metastring tells DOS to display the default drive letter, and $g to show the greater than (>) sign. The characters that can be used after $ are shown in Table 5.1.

Table 5.1 Prompt Metastring Characters

Character	Function
$	$ character
t	Time
d	Date
p	Current directory of the default drive
v	DOS version number
n	Default drive letter
g	> character (greater than)
l	< character (less than)
b	| character (vertical bar)
q	= character (equal)
h	Backspace; to erase previous character
e	Esc
_	(Underscore) Go to beginning of new line

(Note: The letters can be in uppercase or lowercase.)

If a character in Table 5.1 is not preceded with $, it is treated as a literal string and displayed; no action is taken. For example, if you enter this:

```
A>prompt q
```

you get the letter q displayed as the prompt message, not an equal sign.

To display a nonalphabet character appearing in Table 5.1, you must precede it with $. Without $, these characters will generate the error messages shown below:

```
A>prompt >
File creation error
A>prompt <
File not found
A>prompt |
Syntax error
```

These characters have special meanings to DOS, as explained in Chapter 8.

A character other than those listed in Table 5.1 and placed after $ is ignored by DOS, and those after the first character are displayed as a literal string. For example, you could enter this:

```
A>prompt $care
are_
```

On the other hand, you could try this:

```
A>prompt $time
19:05:26.95ime_
```

$t generates time, but the others are treated as a literal string.

Useful Prompt Displays

The metastring list in Table 5.1 contains numerous ways for you to maneuver the DOS prompt. The most useful, however, is the one to show the current directory of the default drive:

```
A>prompt $p$g
```

This command may lead to the display of a new prompt like this:

```
A:\DOS>_
```

Note: If you get lost in your directory maze, just use this prompt as the North Star to navigate around. Moreover, if you are a person who is easily disoriented and constantly needs a road sign to guide you, you can put the above command in your AUTOEXEC.BAT batch file.

If you want to put date and time information in the prompt, this command will do the trick:

```
A>prompt $d $t
```

Notice the space between the two metastrings. Without it, there would be no space separating the date and time numbers shown below:

```
Sun  6-24-1990 11:16:22.30_
```

If you want to display date and time in two lines, you can enter this command:

```
prompt $d$_$t
```

Tips & Tricks

PROMPT $P and a Floppy Drive

If the default drive is a floppy drive, entering the PROMPT $P command alone or in combination with other strings will activate the drive and cause the drive light to appear. If the drive is not ready for reading (e.g., no disk inserted), an error message appears. If you want to change from drive A to B in a one-floppy system, this message also appears:

```
Insert diskette for drive B: and press any key when ready
```

If PROMPT $P is not used or is negated by another metastring, the above message is not shown, nor does the drive light appear.

The prompt message now becomes:

```
Sun  6-24-1990
11:17:48.20_
```

The cursor stays at the end of the second line for you to enter a new command. Notice also that the time information has changed. Every time you make a move, including just pressing Enter, a new time appears, reflecting the latest time.

If the prompt is too long and too intrusive, you may just want to display the time data by entering this command:

```
prompt $t
```

The prompt now becomes:

```
11:19:00.58_
```

Fancy Prompts

What if you want to display only hours and minutes, not seconds and fractions? You can erase the extra characters like this:

```
prompt $t$h$h$h$h$h$h
```

Each $h combination erases a displayed character to the left, just like pressing the Backspace key once; six of them erase six characters. What is left becomes:

```
11:22_
```

Let us now bring back the familiar A> prompt, combined with the current directory and time:

```
prompt $t$h$h$h$h$h$h $p$g
```

A new prompt appears:

```
11:22 A:\DOS>_
```

Now you know not only where you are, but also what time it is each time you enter a new command. From now on, you will no longer be so immersed in learning DOS or using your PC that you lose track of time.

You can put the date-time display at a specific area of the screen and/or in specific colors. To do that, you need to use `ANSI.SYS`. That will be covered in Chapter 11.

Practice 5-4

- ☐ 1. Change the prompt setting to show the default drive, current directory, and the $ sign.
- ☐ 2. Move to three different directories in the directory structure created in the previous Practice session. The screen should show different prompt displays.
- ☐ 3. Change the prompt to show minutes, seconds (no fractions), and the > sign.
- ☐ 4. Restore the original (default) system prompt.

Drill

_____ 13. Used alone, the `PROMPT` command will display a DOS prompt like this:
a. `A:\>`
b. `A:\`
c. `B:\DOS>`
d. `C>`

Answers for questions 14–20:
a. L, b. P, c. N, d. G, e. $, f. D, g. Q.

Each of these characters is used with `PROMPT` and placed after $ for the purpose of displaying:

_____ 14. >

_____ 15. <

_____ 16. =

_____ 17. the dollar sign

_____ 18. the current directory

_____ 19. the default drive

_____ 20. the current date

_____ 21. When you want to erase a displayed prompt character, you use $B to do it. True or false?

_____ 22. The command PROMPT T_PG will display:
a. 9:10:22.12A>
b. 9:10:22.12A:\DOS>
c. 9:10:22.12A:\
d. 9:10:22.12
B:\DIR1>

Using TREE to Display Directories

✎ **Use the external TREE command to display the directory structure of a drive (version 3.3 and before) or a specific directory (version 4).**

The TREE command is used to show the tree structure of your directories. It is useful when your directory structure gets complicated and confusing.

TREE, unlike PROMPT, is an external command. You need to tell DOS where to access the TREE.COM program before you can enlist its help. The TREE command varies significantly between version 3.3 (and those before) and version 4. We discuss them separately below.

Version 3.3

The TREE command of version 3.3 and before can analyze only an entire disk, not specific directories. In its simplest form, it could be executed this way:

```
A>tree
```

This command tells DOS to run the TREE.COM program stored in drive A (unless PATH is used as explained in the next section) and use the program to analyze the disk in drive A, the default drive. If another drive is to be analyzed, you need to specify it, such as:

```
A>tree b:
```

This command tells DOS to analyze the disk in drive B, which is not the default drive.

After running the TREE.COM program, DOS 3.3 or earlier will show a listing like below:

```
DIRECTORY PATH LISTING

Path: \DIR1

Sub-directories:  DIR2

Path: \DIR1\DIR2

Sub-directories:  DIR3
```

```
Path: \DIR1\DIR2\DIR3

Sub-directories:  None
```

The listing shows the directory structure of an entire disk. It shows that DIR1 is a subdirectory of the root directory. DIR1 has a subdirectory named DIR2. DIR2 has DIR3 as a subdirectory. And DIR3 has no subdirectory. From this listing, you can find out what directories and subdirectories you have created and how they are related to one another. To display files too, use the /F (File) switch, such as:

```
A>tree b: /f
```

This command will show all the files in each directory.

Version 4

The TREE command of version 4 has been improved considerably. You can use it to analyze an entire disk or a specific directory. Instead of showing a listing, it displays an easier-to-read diagram. You can still use the /F switch to order the listing of files.

If you analyze a directory, only that directory and its subdirectories are affected. Here is an example:

```
A>tree dir1
Directory PATH listing
Volume Serial Number is 075A-15D2
A:DIR^@
└───DIR2
    └───DIR3
```

Besides the added serial number (no serial number is available if a disk is formatted with earlier versions), the diagram shows the directory structure starting with DIR1. DIR1 has a subdirectory named DIR2, which also has a subdirectory named DIR3. The directories above DIR1 or those not directly related to it are not shown.

If no parameter is entered with the TREE command, the current directory (not the entire default drive) is the target. Suppose the current directory is DIR1. The following appears if TREE is entered with the /F switch, but no parameter:

```
A:\DIR1>tree /f
Directory PATH listing
Volume Serial Number is 075A-15D2
A:.^@
└───DIR2
    │   FILE1.TXT
    │   TESTFILE.DOC
    │
    └───DIR3
```

The single period after the default drive (A:) indicates the current directory. The diagram shows only the directories below the current directory, whose name, by the way, is not shown. The /F switch causes the listing of files, if any, in each directory.

If an entire disk is the target, you may need to specify the root directory. The command below will analyze the entire disk, starting from the root directory of the default drive:

```
A:\DIR1>tree \
```

In version 4, you can also use the /A (ASCII) switch to display the diagram with ASCII characters (instead of drawing lines). This speeds up screen display.

Tips & Tricks

Handling a Large Tree

If you have a large number of files and directories, the listing will continue to scroll off the screen (particularly with version 3.3). To halt the listing before it disappears, you can use the Pause key or Ctrl-Num Lock.

If things get really complicated and you need a printout of your directory structure, you can either press Ctrl-P to echo the display to the printer, or you can redirect the information from the screen to the printer. The tricks you learned in the last section of Chapter 2 can be applied here.

Using PATH to Access Executable Files

- **Use the internal PATH command followed by a list of drives and directories to make DOS search in the list for an executable file.**
- **Use PATH alone to show an existing path list.**

Another useful DOS tool to help you navigate around a complex directory structure is the PATH command. You can use it to make DOS search in predetermined paths for an executable file (with the COM, EXE, or BAT extension) and then execute it.

Suppose you have the BASICA.COM program stored in the root directory or in a directory named BASIC. You now want to run it, but do not know whether it is in drive A or drive B. You can go to each drive and see whether the program is there, and run it after finding it in a specific directory. Or you can try the following:

```
A>path a:\;a:\basic;b:\;b:\basic
A>basica
```

The first line establishes a command path, which can later guide DOS to look for an executable file in the root directory of drive A, then the BASIC directory of drive A, and then drive B. The second command executes the named program.

When DOS encounters the second command, it will search for the BASICA program in the path list of directories in the order they appear. If the program is found, it is executed. If not, you get the familiar error message of:

```
Bad command or file name
```

In that case, the named file is not executable or is not stored in the specified path list. You may then examine what paths DOS is asked to follow. To do that, simply enter this command:

```
A>path
```

DOS will respond by showing you this:

```
PATH=A:\;A:\BASIC;B:\;B:\BASIC
```

That is exactly what you ordered it to do.

Note: If the whole string does not appear, one likely cause is that you have inserted a space somewhere. The existence of a space will cause DOS to terminate reading the rest.

The above display shows that you want DOS to search the root and BASIC directories in both drive A and B. The order can be changed. The important thing is to specify all the drives and directories you want DOS to search. Also, do not forget to separate each path with a semicolon.

You can insert a space after the PATH command, or you can replace it with an equal sign (=). Either one will work. It is more convenient to simply press the space bar.

Each time you enter a new path, the latest path list will replace the old one. DOS remembers only the most recently entered information.

If you want to delete an existing path list, simply put a semicolon after the PATH command, such as:

```
A>path;
```

If you now enter the PATH command alone, DOS displays the No Path message.

A path list, a prompt string, and an APPEND list discussed below are stored in a memory area called the **DOS environment**. You can use the SET command to maneuver them. See Chapter 9 and 11 for more details.

Accessing Data Files with APPEND

- **Use the external command of APPEND followed by a list of drives and directories to make DOS search in the list for a data file.**
- **Use an available switch with APPEND to make DOS search for both data and executable files.**

When you use PATH, DOS will search for executable files (command path), but not data files (data path). You need the APPEND command to search for the latter. If

you use a proper switch, it can search for both data and executable files. It can also be used with a network to access remote data files.

Unlike PATH, APPEND is an external command. But once loaded, it stays resident until power is off. It is known as a **TSR** (terminate and stay resident) or memory-resident program.

Follow these steps to install APPEND and change its parameters:

1. Install APPEND:

   ```
   A>append /e
   ```

   APPEND has now entered the DOS environment.

2. Display the environment variables:

   ```
   A>set
   ```

   You may see other things, but not APPEND.

3. Set APPEND parameters:

   ```
   A>append a:\;a:\dos;b:\;b:\bat /x
   ```

   We are including a path list, plus specifying the /X switch whose purpose is explained below.

4. Display APPEND's list:

   ```
   A>append
   APPEND=A:\;A:\DOS;B:\;B:\BAT
   ```

You can now use APPEND as you do PATH as explained earlier. If you enter the SET command now, the above list will also appear.

Suppose you now enter this command:

```
A>type x.bat
```

DOS will search in all the drives and directories in the specified list to display the first matching file.

APPEND has these switches (version 3.3 has only /E and /X):

/E	Store paths in DOS environment; can then be changed by the SET command
/PATH:ON	Search both command line and environment paths
/PATH:OFF	Ignore environment path if command line path used
/X or /X:ON	Act like PATH, search for COM, EXE, BAT
/X:OFF	Does not search for executable files
;	Placed after APPEND to clear path list

The /E switch causes APPEND to be stored in the DOS environment, like PATH and PROMPT. You can then manipulate it as an environment variable, as explained in Chapter 9. If you want to put it in the environment, you must specify this switch without any path list when you install APPEND (running it the first time). If you do it after that, you will get the `Invalid switch` message. This is true of both versions 3.3 and 4.

The /X: switches instruct DOS whether or not to search for executable as well as data files along the path list. Use /X or /X:ON to turn on this feature, and /X:OFF to turn it off.

Version Variances: In version 3.3, you can use /X only when you install APPEND.EXE. Furthermore, the /X:ON and /X:OFF switches are not available. That means once installed with /X, the switch can no longer be disabled.

When the /X switch is on, you can use COPY to search for source files. Suppose you are in A:\ and FILE1 and FILE2 are stored in the A:\FILES directory. You can enter this command:

```
A:\>copy file? b:
```

COPY will locate the source files. Other commands, such as DIR, XCOPY, PRINT, and REPLACE, will not find the matching files. If /X is not turned on, even COPY will not find these files.

The /PATH: switches (version 4 only) tell DOS whether or not to search beyond the path of a file entered on the command line. If set on (the default) by this command:

```
A>append b:\bat /path:on
```

and if you enter this command:

```
A>type b:\x.bat
```

DOS will go to the root directory of drive B (the path specified on the command line), plus the B:\BAT directory (specified in the APPEND list), to search for X.BAT. If set to off, DOS will only search in drive B's root directory, as specified on the command line. If no path is specified on the command line, such as:

```
A>type x.bat
```

DOS will search all the paths in the APPEND list regardless of whether PATH is on or off. If a matching file is found in the current directory, it has priority over those in the list.

Warning: APPEND can cause confusion. When you retrieve a file, DOS may find it somewhere. When you resave the file, it goes to the current directory, instead of the original one. For example, if you enter EDLIN (explained in Chapter 11) with this command:

```
A>edlin x.bat
```

X.BAT may be retrieved from the B:\BAT directory. When you exit, the revised version will be saved to drive A and the original one will not be changed. One way to avoid this confusion is to specify this switch:

```
A>append /path:off
```

and then provide a specific path before a file name:

```
A>edlin b:\bat\x.bat
```

This will retrieve a file from, and save it to, the specified path and nowhere else.

Tips & Tricks

The Hazards of a TSR Program

A TSR (terminate and stay resident) or memory-resident program stays in the memory until power is off. Once loaded, it can be activated or made dormant at will. You can then run another program, such as a word processor.

There are many such commercial programs available today. The most popular is Borland's Sidekick. They are popular with computer users, for they can perform many small but useful functions not available in major application software.

There is so far no set standard as to where each TSR program should stay in your computer memory. If you use some of these, they may claim the same memory area. If that happens, your computer will be frozen, requiring rebooting.

APPEND, like other TSR programs, could clash with other programs in memory. If I run WordPerfect's Shell program and then run WordPerfect, I cannot go to DOS, run APPEND, and return. The computer will simply freeze. On the other hand, if I run APPEND first and then load Shell and WordPerfect, I have no problem. The order in which you run a TSR program could be a matter of life or death.

Practice 5-5

- ☐ 1. Use the TREE command to show the directory structure of a drive other than the one storing TREE.COM.
- ☐ 2. Use TREE with the switch that will show files.
- ☐ 3. If you have version 4, use TREE to display the directory structure of DELTA, created in Practice 5-3.
- ☐ 4. Enter a PATH list that would allow you to execute from anywhere a program that is stored in the DELTA directory. Display the path list on the screen.
- ☐ 5. Use APPEND with the switch that will make DOS search for both data and executable files in the DELTA directory. Display the APPEND path list on the screen.

(Remove the directory structure created in Practice 5-2.)

Drill

_____ 23. TREE, APPEND, and PATH are all external commands. True or false?

_____ 24. In DOS version 4, the command `A>tree b: /f/a` will:
a. Analyze drive A.
b. Use ASCII characters and display files.
c. Display only ASCII files.
d. Cause an error message due to an illegal switch.

_____ 25. In version 3.xx, TREE can be used to analyze a whole disk but not a specific directory. True or false?

_____ 26. Used alone, PATH tells DOS to clear the previously entered path specifications. True or false?

_____ 27. You can use PATH to search for a nonexecutable file. True or false?

_____ 28. If you enter PATH followed by specifications several times, only the most recently entered information will be remembered by DOS. True or false?

_____ 29. APPEND is an external command, but it stays in the memory after it is used the first time. True or false?

_____ 30. APPEND can be used to access both data and executable files. True or false?

Summary and Review

Check each item as you review and understand the following highlights of the chapter:

☐ 1. Every disk has a root directory. You can, if you wish, create other directories and subdirectories. The root directory is the highest-level directory. Other, lower-level directories can be structured in any way you wish.

☐ 2. The current directory is the active directory, where actions can be taken without giving DOS a path. The directory above (one level higher) is referred to as the parent directory. The one below (one level lower) is referred to as a subdirectory. The granddaddy of all directories is the root directory.

☐ 3. Using directories to store a large number of files has many advantages. You can better organize and find your files if you store different types in different directories. For example, letters can be in one directory and reports in another.

- ☐ 4. Use the MD (or MKDIR) command followed by a name to create a directory that is a subdirectory of the current directory. If the directory to be created is not a subdirectory of the current directory, provide a path before the directory name to be created.
- ☐ 5. Use the RD (or RMDIR) command followed by a name to remove a directory. Provide a path if the directory to be removed is not a subdirectory of the current directory. A directory must be empty (not contain anything) before it can be removed.
- ☐ 6. Use the CD (or CHDIR) command to change the current directory. Put a name after the command if you want to move to a subdirectory. Put two periods (..) after it if you want to move to the parent directory. Add a specific path for others.
- ☐ 7. When actions related to the current directory are to be taken, just enter a command such as DIR. If these actions are related to other directories, you need to add a path.
- ☐ 8. Take advantage of the PROMPT command to display meaningful messages. If you want the system prompt to show the name of the current directory in the default drive, use the PROMPT PG command.
- ☐ 9. A number of metastrings can be put after the PROMPT command to perform some useful tasks, such as showing the current time.
- ☐ 10. If your directory structure becomes complicated and confusing, use the external TREE command to show it. Version 3.3 or earlier shows a listing and can analyze only a whole disk. Version 4 shows an easier-to-read diagram instead of listing and can be used to analyze a specific directory.
- ☐ 11. Use the PATH command followed by drive and directory specifications to establish a command path, which can guide DOS to search for an executable file when you enter the program name to execute it. This can save the time and trouble of looking for it manually. Use the PATH command alone to display the most recently entered path list.
- ☐ 12. Use APPEND to establish a data path, which can guide DOS to access data files that may be stored in places other than the current directory. APPEND stays in the memory after running the first time. This could conflict with other programs and crash your computer.

Review Questions

Write an answer for each question below.

1. What are some of the advantages of using directories to save files?
2. Define root directory.
3. Define current directory.

4. What happens if you enter this command:

```
A>md fatcat
```

5. Enter a command at the prompt below to create a directory named RUNNER, which is a subdirectory of the root directory:

```
A:\SPORTS\SHOPS>_
```

6. How does DOS interpret this command:

```
A:\SPORTS>rd shops
```

7. What is the easiest way to designate the directory directly above the current one as the new current directory?

8. Enter a command below to make the root directory the current directory:

```
A:\SPORTS\SHOPS>_
```

9. What happens if you enter the PROMPT command alone?

10. If you want the prompt to show the current directory of the default drive plus the greater than (>) sign, what command do you enter?

11. What command do you use if you want the prompt to display the current date followed by the default prompt in a new line?

12. What prompt message will appear if you enter this command:

```
PROMPT=$Greetings, world:
```

13. What does the following command do:

```
A>c:\dos\tree
```

14. What is the purpose of using PATH or APPEND followed by some specifications?

15. What is the purpose of using the PATH or APPEND command alone? How do you clear a path list?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
TREE	Shows list, not graphical diagram	Shows graphical diagram
	Can only analyze a whole disk	Can analyze disk and directory
	No serial number	Shows serial number
		New switch: /A
APPEND	To search for executable files, the /X switch must be specified when APPEND is installed; once installed, it can't be disabled	/X need not be installed, can be used any time
		New switches: /X:ON, /X:OFF, /PATH:ON, /PATH:OFF

Copying and Comparing Files

Chapter 6

TOPICS:

The COPY Command's Many Uses
Combining (Concatenating) Files
Using Wild Cards to Combine Files
Changing a File's Date-Time
Summary of Combining Files
The COPY Command's Switches
Copying to Devices
Using the XCOPY Command
The XCOPY Command's Switches
Copying a Whole Disk with DISKCOPY
Comparing COPY, XCOPY, and DISKCOPY
Copying Copy-Protected Software
Comparing Files with COMP
Comparing Files with FC
Comparing Disks with DISKCOMP

COMMANDS COVERED:

COPY, VERIFY, XCOPY, DISKCOPY, COMP, FC, DISKCOMP

NEW TERMS AND SYMBOLS:

+, ^Z, →

REQUIRED MATERIALS:

1. Two practice disks
2. Access to COMP.EXE (PC-DOS) or FC.EXE (MS-DOS), DISKCOPY.COM, DISKCOMP.COM, and XCOPY.EXE

Quick Start

✎ **To combine and copy to the last file, which will be created or replaced:**

```
A>copy file1 + file2 fileall      (or)
A>copy file* fileall
A>copy /b prog* prog.all   (binary mode)
```

✎ **To append (add) to the first file, which must exist:**

```
A>copy file1 + file2 + file3      (or)
A>copy file1 + file?
A>copy /b prog1 + prog2    (binary mode)
```

✎ **To change file dates:**

```
A>copy file . +          (single file)
A>copy file? . +         (matching files)
A>copy . . +             (all files—ASCII mode)
A>copy /b *.exe . +      (binary files)
```

✎ **To copy to devices:**

```
A>copy con prn           (from console to printer)
A>copy file prn          (from disk to printer)
A>copy file con          (from disk to screen)
A>copy file? con         (display multiple files)
A>copy /b *.exe con      (display multiple binary files)
```

✎ **To copy files and subdirectories with prompt for `Y/N`:**

```
A>xcopy /s/p a:file? b:
```

✎ **To compare files:**

```
A>comp a:file?.ext b:*.bak       (PC-DOS)
A>fc file1 file2                 (MS-DOS)
```

✎ **To duplicate a floppy disk track by track:**

```
A>diskcopy a: b:
```

✎ **To compare two floppy disks:**

```
A>diskcomp a: b:
```

Copying files is a vital part of working with a computer. The files you acquire or create need to be protected with backup copies; if one is corrupted, you still have the backup. Copies can also be distributed to other persons or used on other computers. All these options would not be possible if you could not make copies.

So far you have learned to use the `COPY` and `DISKCOPY` commands to produce duplicates. In this chapter you learn to use `COPY` to do more tricks, including combining files and putting a new date and time on existing files. The `XCOPY`

command, an improved version of COPY, is also covered in this chapter. Some of its functions related to backing up files will be explained in the next chapter.

The three commands, COPY, XCOPY, and DISKCOPY, overlap in some aspects but differ in others. You will find out how they differ and how to apply them to different situations. After learning different ways of copying files, you will also learn to compare files and disks to find out possible variations and identify errors.

The COPY Command's Many Uses

✎ **COPY's uses, explained before, are summarized here.**

You have learned different ways to use the COPY command. The list below sums up what we have covered on this commonly used command:

Command	Purpose
A>**copy filename b:**	Copy one file from A to B
A>**copy b:*.***	Copy all files from B to A
A>**copy b:.**	Same as above
A>**copy b:file?.txt**	Copy matching files from B to A
A>**copy chap*.* b:*.bak**	New extension in target files
A>**copy chap*.* *.bak**	Same directory, new extension
A>**copy con filename**	Create a file and save in A
A>**copy filename + con**	Add to an existing file in A
A>**copy filename con**	Display a file on the screen
A>**copy file?.txt con**	Display matching files

The COPY command follows this syntax:

```
A>copy source target
```

If there is only one parameter, as in the second and fourth examples above, it is assumed to be the source. The target is assumed to be the current directory of the default drive.

These examples copy specific or matching files from the source to the target with or without changes in the file names. Duplicate files carry the date-time of the original files. What if you want the duplicate files to have a different date-time, or want to combine several files into one, or want to handle binary files or copy to devices? The sections below explain these possible variations.

Combining (Concatenating) Files

✎ **Append all the files to the first by connecting all the files with a +.**

✎ **Combine files with a + and copy them to the last file, which must be separated from the rest with a space.**

The COPY command can be used to combine several files into one. Although a target file includes everything from source files, the latter remains intact.

Let us create two separate files and try to put them into one. Insert a practice disk in drive A and follow these steps:

1. Create the first file:

```
A>copy con file1
This is from FILE1.
^Z
```

2. Create the second file:

```
A>copy con file2
This is from FILE2.
^Z
```

3. Combine FILE1 and FILE2 into FILE3:

```
A>copy file1 + file2 file3
FILE1
FILE2
        1 File(s) copied
```

Notice the absence of the + before the last file. This plus sign plays a crucial role, as you will see in the examples below. In this command, DOS treats the files connected with a + as source files and the last, not connected with a +, as the target. The space before and after the + is optional.

Note: The screen displays the message `1 File(s) copied`. This actually means that one file has been created, namely FILE3. If this file does not already exist, it will be created. If it already exists, its original contents will be replaced.

4. Display the contents of the combined file:

```
A>type file3
This is from FILE1.
This is from FILE2.
```

Notice the order: FILE1 is placed before FILE2. This order is determined by the way you arrange the files in step 3.

So far both FILE1 and FILE2 remain unchanged. You can use the TYPE command to show their contents. The following steps, however, will change their contents.

5. Show the creation date-time of FILE2:

```
A>dir file2
```

Write down (or memorize) the time of FILE2's creation. This number will be compared to a new time shown later.

6. Combine all three files:

```
A>copy file2 + file1 + file3
FILE2
FILE1
FILE3
        1 File(s) copied
```

After you enter the command, three file names are automatically displayed. Again, you are told that one file has been copied, which you should interpret to mean that one file, namely FILE2, has been created.

Note: Notice the order of the files displayed on the screen. Since all the files in the command are connected with a +, you ask DOS to append to the first file. FILE2, which is placed at the beginning of the list, is the target, and the rest are sources. The target file is changed, but the source files remain intact.

7. Show the contents of the target file:

```
A>type file2
This is from FILE2.
This is from FILE1.
This is from FILE1.
This is from FILE2.
```

The first line comes from the original FILE2, the second from FILE1, and the last two from FILE3.

8. Show FILE2's creation date-time:

```
A>dir file2
```

Compare this time to the one shown in step 5. You can now safely conclude that DOS treats FILE2 as a new file with a new date and contents.

What about FILE1 and FILE3, placed after the first + sign? They remain unchanged, keeping the same contents and date-time data. You can prove it by using the TYPE command on them. You can append from as many files or from as many directories as you wish. Consider the following example:

```
A>copy file1 + b:file2 + c:\dosbook\chap3 + chap4
```

After this command, FILE1 not only will keep its original contents but also will include the contents of all the other source files shown after the first +; all the latter files remain unaltered.

Warning: If you link a series of files with a + to append to the first, you should be aware of an unusual situation. If the target (first) file does not exist, the next existing file will become the target. Consider this:

```
A>copy file0 + file1 + file2 + file3
```

If FILE0 does not exist, FILE1 will be the target and the others will be appended to it.

Using Wild Cards to Combine Files

- **Combine all the source files matching a pattern and copy them to a target, which must be separated with a space.**
- **Append to a target placed before the + all the source files matching a pattern placed after the +.**

Wild card characters can be used in combining files. Several related files can be combined and put in a new file, such as:

```
A>copy file*.txt fileall.txt
```

In this example, `FILE1.TXT`, `FILE2.TXT`, etc. are copied to `FILEALL.TXT`. If `FILEALL.TXT` does not exist, it will be created. If it already exists, its contents and date will be replaced.

A command like the one above will generate a peculiar situation and this cryptic error message:

```
Content of destination lost before copy
```

When this command is executed, DOS goes through a series of matching files, such as `FILE1.TXT`, `FILE2.TXT`, etc. and copies each to `FILEALL.TXT`. The matching files will also include the target file itself, namely `FILEALL.TXT`. The error message is caused by the earlier version of `FILEALL.TXT` being replaced by a later version. Copying, however, continues with the next matching file, if any.

Instead of copying related files, you could append them, such as:

```
A>copy file1.txt + file*.txt
```

This will also generate an error message, as shown above. However, all the files matching the `FILE*.TXT` pattern will be appended to the target file (located before the +), except the target file itself. If the target file does not exist, no copying will be done; a series of error messages will appear instead.

Note: In either example above, do not use a wild card in the extension of the target file, such as `FILEALL.*` or `FILE1.*`. If you do, only the last matching source file will be copied to the target. The ending message on the screen shows multiple files copied, not just one. When you see this message, you will know that file concatenation has not been done.

Using wild cards to copy matching files to a combined file can produce unpredictable results, depending on the order of the file names stored in the directory. As an illustration, let us suppose that you have created three files named `F1`, `F2`, and `F3` in that order (the `DIR` command should show this order), and that each file contains the content of 1, 2, and 3 respectively. Now consider these two commands:

```
A>copy f? f1
A>copy f? f2
```

The first command results in `F1` having the contents of "123" and shows no "Content of destination lost before copy" message. It produces the same effect as:

```
A>copy f1 + f2 + f3
```

The second command results in F2 having only "13" and a display of this message:

```
A>copy f? f2
F1
F2
Content of destination lost before copy
F3
        1 File(s) copied
```

Any target file whose name is not located in the directory ahead of all the other matching files will have its contents wiped out in such an operation, as the original F2 file in our example. On the other hand, A>copy f2 + f? will result in F2 having 213, even though the same error message appears.

Changing a File's Date-Time

✎ **Place a + after a file name to put the current date-time on the file. Use wild cards for matching files.**

The COPY command can be used to put the current date and time on an old file. This will make an old file appear to be created more recently, or earlier, if you set your computer time to an earlier date. The following is one way to do it:

```
A>copy file*.txt b:*.* +
```

This copies all the matching files to drive B, keeping the same file names but using the current date and time.

Warning: The + should be put at the end of target files, not source files. If you put it after the source, DOS will interpret your instruction quite differently. Consider the following:

```
A>copy file1 + b:
```

It appears that you intend to copy a file from A to B and stamp the current date and time on the new file. DOS, however, will append all the files in the current directory of drive B to the file before the +.

If there is a single file on which you want to stamp a new date and time, try this:

```
A>copy file1 + ,,
```

Warning: The above example, appearing in IBM manuals, is hazardous. Do NOT use a wild card character in such an operation, such as:

```
A>copy file? + ,,
FILE1
File not found -
Content of destination lost before copy
        0 File(s) copied
```

FILE1, the first matching file found by DOS, is now deleted. The message tells you that the file can no longer be found.

If you append an existing file to a nonexistent file, or vice versa, the existing file will also carry a new date and time while keeping its original name and contents, such as:

```
A>copy file1 + dummy
```

If DUMMY exists, it will be appended to FILE1. If not, FILE1 will carry a new date and time. The order of the two files can be reversed without affecting the outcome. Again, do not use wild cards in the target (before the +); the first matching file will also be lost as explained above.

Tips & Tricks

Changing File Dates

There are many ways to change file dates. The simplest and safest way, which I stumbled upon while testing `COPY`'s various combinations, is to use a single period judiciously, such as:

```
A>copy file1 . +
A>copy file*.txt . +
```

The first command will change the date-time of one file, and the second of all the matching files. The space after the period is optional, but the one before it is mandatory. Without this space, DOS will consider your move to be copying a file to itself and will refuse to do anything.

If you want to change all the files in one directory, try either command below:

```
A>copy *.* *.* +
A>copy . . +
```

As explained in Chapters 4 and 5, a single period can be used in lieu of `*.*`. The above two commands are thus identical.

If you have multiple files to update but want to keep them on the current disk, try the following:

```
A>copy file*.txt *.* +
```

Every time a file is worked on, the message `Content of destination lost before copy` will be displayed. Each target file is replaced with a matching source file. Thus the contents of each file remain unchanged. However, since each target file is created by this operation, new date and time data is attached to it.

Summary of Combining Files

✎ **Rules for combining files are summarized here.**

The rules for combining files may seem complicated. Perhaps the summary examples below may help:

Append to First File:

A1. `A>copy target + source1 + source2...`
A2. `A>copy all.txt + *.txt`

Combine and Copy to Last File:

```
B1. A>copy source1 + source2... target
B2. A>copy *.txt all.txt
```

Change an Existing File's Date-Time:

```
C1. A>copy file1.txt + ,,
C2. A>copy file1.txt . +
C3. A>copy file1.txt + dummy
C4. A>copy dummy + file1.txt
C5. A>copy file*.* *.* +
C6. A>copy b:file*.* +
C7. A>copy *.* *.* +
C8. A>copy . . +
```

The A examples append all the files (listed or matching) to the file appearing before the first +. In such cases, where all the files are connected with a +, the first file is the target and all the others are sources. In A1, if the first file does not exist, the first existing file in the list is treated as the target. In A2, if the target file does not exist, no copying is done.

The B examples show all the source files connected with a +, the target file appearing last and separated from the rest with a space. All the source files go to the target, which is created if nonexistent or completely replaced if existent.

Tips & Tricks

Hazardous COPY Parameters

When used with certain parameters, the COPY command could erase your files. Consider these examples:

```
A>copy file? combo
A>copy file? + ,,
A>copy file? + dummy
A>copy file? filex
```

The first will replace the contents of COMBO if it exists. The second will delete the first matching file found and do nothing to the other files, if they exist. What the third example does depends on whether DUMMY exists. If it does, it is appended to the first matching file. If not, the first matching file is deleted, just as in the second example. (If FILEX is not the first file in the directory matching FILE?, its original contents will be lost.)

If you want to stamp the current date on a binary file, make sure you use the /B switch:

```
A>copy /b prog.exe + ,,
A>copy /b prog*.com . +
```

Without the /B switch (the space before the / is optional), your updated programs would be much shorter, only up to the first ^Z; they would most likely be ruined by such an operation. See the next section for an explanation.

The C examples show various ways of updating (adding a new date and time to) old files. C1–C4 update a single file in the current directory. C5 updates all the matching files in the current directory. C6 copies matching files to another drive, updating the target files. C2–C5 will also generate error messages without hindering the updating. Finally, C7 and C8 let you change all the files in one directory.

Practice 6-1

- ☐ 1. Create two files named `TEST1` and `TEST2`, each containing one line of text.
- ☐ 2. Combine `TEST1` and `TEST2` into a new file named `TEST3`.
- ☐ 3. Append `TEST1` and `TEST2` to `TEST3`; link all three files with a +.
- ☐ 4. Append `TEST2` and `TEST3` to `TEST1`; use wild cards.
- ☐ 5. Change all the files to the current date-time.
  (Keep these test files until you are instructed to delete them.)

Drill

B 1. The command `A>copy file??.* *.bak` will copy:
   a. matching files to another disk
   b. matching files to the same disk with new extensions
   c. only matching source files that have no extensions
   d. nothing; it's illegal

C 2. The command `A>copy filename +b:` will:
   a. copy a file to drive B
   b. copy a file from drive B to drive A
   c. append files from drive B to the named file
   d. do nothing; it's illegal

B 3. Which of the following commands is legal?
   a. `A>copy *.* a:`
   b. `A>copy *.* b:`
   c. `A>b:copy *.*`
   d. `A>b:copy *.* a:`

C 4. The command `A>copy filename +,,` will:
   a. append all the matching files to the named file
   b. copy the file to the default drive
   c. put the current date and time on the file
   d. do nothing

D 5. The command A>copy file1 + file2 + file3 will:
   a. append the first two files to the third
   b. create the third file if nonexistent
   c. replace the third file if it exists
   d. append the last two files to the first

D 6. The command A>copy file1 + file2 file3 will:
   a. append the first two files to the third
   b. create the third file if nonexistent
   c. replace the third file if it exists
   d. do all of the above

The COPY Command's Switches

✎ **Use /V to verify writing, /A for ASCII mode, and /B for binary mode.**

✎ **Use /B when putting a new date-time on a binary file.**

The COPY command comes with three switches:

/V	Verify target files
/A	Copy ASCII files
/B	Copy binary files

Verifying Writing

The /V switch turns on VERIFY for a particular copying operation. For example, this command:

```
A>copy file1 b: /v
```

copies FILE1 from drive A to drive B and verifies the target file's writing.

The above example turns on the VERIFY function only during this particular operation. The function is off in future copying operations, unless you use the /V switch again.

You can turn on VERIFY by entering this command:

```
A>verify on
```

After this command, every copying operation will be followed by verification. Adding the /V switch after this does not change its status, but the switch is redundant.

To find out whether VERIFY is on or off, you can enter the command alone:

```
A>verify
VERIFY is on
```

DOS responds by showing the status. To turn VERIFY off, this command will do:

```
A>verify off
```

Entering VERIFY now shows the off status:

```
A>verify
VERIFY is off
```

The VERIFY function is rarely needed because the act of copying a file seldom goes wrong. If you want to be doubly sure, it causes no harm to use it. The only thing you lose is time—it takes extra time to check a target file.

ASCII or Binary Mode

The /A and /B switches tell DOS to be in ASCII or binary mode during copying or combining files.

When DOS copies a file, it operates by default in binary mode. Everything in the source is faithfully read and then written to the target.

When combining files (including any operation using a +), however, COPY operates by default in ASCII mode. Each source file is read only before the first ^Z character encountered; the rest is ignored. The target file also has an extra ^Z character added to the end.

The ^Z Character

The ^Z character is a remnant of the past. Before version 3.0, DOS used this character (ASCII 26) to identify the end of a file. Newer versions use the value stored in the disk directory to determine each file's size. These versions can handle files without the ^Z, but still honor it when it is encountered. If a file contains no ^Z or has a ^Z at the very end, DOS knows its size and can handle it without any problem.

However, if there is a ^Z in the middle of a file, as in many binary files (such as computer programs), this may create problems. If you want to combine such files, you may need to use /A or /B switches selectively.

When DOS encounters COPY used with the /B switch in the source, it will read everything. But the reading will end at the first ^Z when /A is used. In case of a target file, /A causes the addition of ^Z to the end (unless it already exists), but /B does not.

Another rule to remember is that the /A or /B switch affects the file immediately preceding it and all the files after it—unless negated by a later switch.

The following examples provide ample illustrations and explanations for the rules stated above:

A>copy fil* b:	Read all in source, no extra ^Z in target
A>copy fil b:/a	Read all in source, add ^Z in target
A>copy fil/a b:	Read before first ^Z, add ^Z in target
A>copy fil/a b:/b	Read before first ^Z, no extra ^Z in target
A>copy fil* b:all	Read source before ^Z, add ^Z to ALL
A>copy f1+f2+f3	Read F2 & F3 before ^Z, add ^Z in F1
A>copy con fil	No ^Z added to FIL
A>copy/a con fil	Add ^Z to FIL
A>copy fil+con	Add ^Z to FIL
A>copy fil+,,	ASCII mode, add ^Z to FIL
A>copy/b fil+,,	Binary mode, no extra ^Z added
A>copy fil .+	ASCII mode, no extra ^Z added
A>copy fil con	Display before first ^Z, same as TYPE
A>copy/b fil con	Display all, including ^Z

As you can see, when simple copying is involved, binary mode is the default unless you add switches.

Warning: When file combination is done, the reading of a source file stops at the first ^Z encountered and an extra ^Z is added to the end of the target file. This also applies when you try to stamp a new date and time on an existing file. If you did not use /B with a binary file, an updated binary file would most likely be incomplete and useless.

The last items in the above list, copying to the console, will be explained more fully in the next section.

Copying to Devices

✎ **Use COPY to copy data to PRN and CON. Use /B with CON to display binary files.**

The COPY command can be used to communicate with devices such as the console (keyboard and monitor), printer, modem, and so on.

In Chapter 4, you encountered the COPY CON command, used to copy from the console to a file or vice versa. Other devices can be used in a similar manner. This technique can be used to communicate with the printer.

Copying to the Printer

To demonstrate how you can send data to the printer, follow these steps:

1. Enter the command to copy from the console to the printer:

```
A>copy con prn
_
```

2. Type the following line and press Enter:

```
This is a test.
```

3. Turn on the printer, if it is not already on.
4. Enter the ^Z character (press F6 and Enter). The screen appears as below:

```
A>copy con prn
This is a test.
^Z
        1 File(s) copied
```

The single line of text is sent to the printer and the print head returns to the left margin. If ^Z is placed at the end of the text line, the print head will stay in that position rather than returning to the left margin.

When you give the COPY CON PRN command, you instruct DOS to copy from the device named CON to the device named PRN. PRN can be replaced with LPT1. If you want to send data to the second printer, you need to use LPT2.

Another way to use COPY to communicate with the printer is to send an existing disk file to it, such as:

```
A>copy file1.txt prn
        1 File(s) copied
```

If the copying is successful, DOS tells you that one file has been copied. If the file cannot be found, this message appears:

```
A>copy file5 prn
FILE5 File not found
        0 File(s) copied
```

If the printer is not on, the computer freezes and you get this response after a little pause:

```
A>copy file3 prn
Write fault error writing device PRN
Abort, Retry, Ignore, Fail? _
```

If you press A, the DOS prompt reappears and the previous command is aborted. If you turn on the printer and press R, the text is printed. If you choose I or F (versions before 3.3 have no Fail option), the temporary freeze continues until the same error message reappears.

Copying to the Console

You can use COPY to copy a disk file to the screen. That has the same effect as using the TYPE command. One major difference is that COPY can combine several files and display them on the screen. Consider this command:

```
A>copy file1 + file2 con
```

All the contents of the first file, followed by the second, will quickly scroll through the screen. At the end DOS tells you: `1 File(s) copied.`[1]

When used to copy from a disk file to the screen, COPY operates in ASCII mode. It stops reading when the first ^Z is encountered. This works the same as using TYPE. Although TYPE does not allow a switch, you can use the /B switch with COPY CON to read an entire file, such as:

```
A>copy file1 /b con
```

This is the trick you can use to see whether a file contains the ^Z character. Where there is such a character, it appears on the screen as a right arrow sign (→). This sign has the value of ASCII 26, which is also the value of ^Z.

You can also use wild card characters when copying files to the screen, such as:

```
A>copy file? /b con
```

All the matching files will be displayed one after another until the end. You can use Pause or Ctrl-S (or Ctrl-Num Lock) to halt the continuous scrolling.[2] A file created with COPY FILENAME + CON (appending from the console to an existing file) will add ^Z to the end of the file. You can use COPY /B FILENAME +, to get rid of it.

[1] This technique can also be applied to PRN. Several files can be combined and sent to the printer. The /A and /B switches can be freely used. Here is an example:

```
A>copy file1 /b + file2 /a prn
```

The entire FILE1 will be read and combined with FILE2's contents before the first ^Z, and the combined file will be printed.

[2] While experimenting with this device, I discovered a peculiar pattern not documented anywhere. If you use the /B switch to display a long file, you cannot break the scrolling. If you have a long file and a slow computer, this simple action could occupy the computer for a long time. On the contrary, the scrolling can be aborted (using Ctrl-C or Ctrl-Break) when no switch is used.

Practice 6-2

- ☐ 1. Use `COPY`'s binary switch to individually display the three test files created earlier, and determine which file contains the `^Z` character.
- ☐ 2. Use `COPY` and wild cards to display all three files one after another.
- ☐ 3. Combine and print `TEST2` and `TEST3`; use a single command for both actions.

Drill

D 7. The command `B>copy file1 + file2 prn` will:
   a. combine two files and send them to the screen
   b. append `FILE2` to `FILE1`
   c. append `FILE1` to `FILE2`
   d. combine two files and print them

D 8. The command `A>copy file?? fileall` will:
   a. read source files before `^Z` and add `^Z` to the target
   b. create `FILEALL` if it does not exist
   c. replace the contents of `FILEALL` if it exists
   d. do all of the above

C 9. Which of the following commands reads only up to the first `^Z` before copying?
   a. `A>copy filename b:`
   b. `A>copy filename /b b:`
   c. `A>copy filename /a b:`
   d. `A>copy a:filename b:`

F 10. When copying files, the `COPY` command operates by default in ASCII mode. True or false?

D 11. Which command will not add `^Z` to the target file?
   a. `A>copy filename /a b:`
   b. `A>copy filename b: +`
   c. `A>copy b:filename /a`
   d. `A>copy filename /a b: /b`

T 12. The two commands below behave exactly alike. True or false?
```
A>type filename
A>copy /b filename con
```

Using the XCOPY Command

✎ **Use XCOPY to speed up copying large files and duplicate subdirectories.**

The XCOPY command first appeared in version 3.2. It is an improved version of COPY. It has three major properties:

✎ **It works like COPY, except faster.**

✎ **It creates and copies subdirectories.**

✎ **It copies files with the A attribute.**

The first two characteristics are covered below. The third, used for backing up files, will be covered in the next chapter. You will better understand its use after studying file attributes, also covered in the next chapter.

Because XCOPY is more than 17KB long (version 4.01), it would take too much memory to put it in the COMMAND . COM file. Consequently, to use XCOPY you need to provide the path to this external command.

The syntax of XCOPY is the same as for COPY:

```
XCOPY source target /switch
```

The following is an example:

```
A>xcopy chap*.txt b:
```

This command will load XCOPY . EXE from drive A (unless you have used the PATH command to tell DOS where to find it) and then copy all the matching files from drive A to B . XCOPY first reads all the matching files to the available memory and then starts copying. The screen displays each file's name as it is copied:

```
A>xcopy chap*.txt b:

Reading source file(s)...
CHAP1.TXT
CHAP2.TXT
CHAP3.TXT
     3 File(s) copied
```

XCOPY will execute your instruction without hesitation under most circumstances. However, one particular circumstance could cause confusion—copying a single file to the same directory, such as:

```
A>xcopy chap1.txt chap1.bak
```

There are two possibilities with this command. One is to copy the first file to the second file, creating it if nonexistent or replacing it if existent. The second is to copy the first item, which is a file, to the second item, which is a subdirectory to be created, if necessary.

When COPY encounters this situation, it will copy the first file to the second, if the second is an existing subdirectory. If such a subdirectory does not exist, the second item will be treated as a file name.

Since XCOPY can create a subdirectory before copying files, it needs to know your intention. Hence this question:

```
A>xcopy chap1.txt chap1.bak
Does CHAP1.BAK specify a file name
or directory name on the target
(F=file,  D=directory)? _
```

If you press F, CHAP1.TXT will be copied to the file named CHAP1.BAK. If you press D, another copy of CHAPT1.TXT will go to the subdirectory named CHAP1.BAK, if it exists; if not, a new subdirectory will be created before copying.

Do not be surprised if DOS fails to create a subdirectory under this circumstance. If a file bearing the same name exists or if you use a DOS device name such as PRN, no subdirectory will be created and no copying will be done. The next section elaborates on this.

The XCOPY Command's Switches

✎ **Use XCOPY's switches for various copying purposes.**

✎ **Do not use XCOPY to combine files or send data to DOS devices.**

The XCOPY command has quite a few switches, as shown below:

/A	Archive, copy files marked with A
/D:mm-dd-yy	Date, copy files on or after the specified date
/E	Empty target directories also copied, used with /S
/M	ReMove source archive bit after backup
/P	Prompt for Y/N when wild cards are used
/S	Subdirectories with files are copied
/V	Verify
/W	Wait for keyboard action before copying

The /V switch looks familiar. In fact, it serves the same purpose as the identical switch used with COPY—to verify target files.

Copying Subdirectories

The /S switch is not only useful but also powerful. It is used to copy matching source files in the current directory and its subdirectories. If an identical directory structure does not exist in the target, it will be created before copying. Consider this command:

```
C:\>\dos\xcopy *.bat a: /s
```

It goes to the DOS subdirectory to load XCOPY.EXE and copies all the matching files from the current directory (which happens to be the root directory in this case) down. The screen may show this:

```
C:\>\dos\xcopy *.bat a: /s
Reading source file(s)...
AUTOEXEC.BAT
DOS\DING.BAT
ARAB\NIGHT\SING.BAT
     3 File(s) copied
```

Notice that the full path of each matching file is shown on the screen. The same pattern will be duplicated in the target.

Now go to drive A and explore what the command has done. The directories, if they did not already exist, are created. And the files displayed on the screen are copied there.

The /S switch is powerful, but it cannot work magic. Suppose you have batch files located in this path:

```
C:\BAT
```

Suppose you are now in the root directory and want to copy the batch files to another subdirectory (C:\DOS). You may consider using this command:

```
C:\>xcopy *.bat \dos /s
```

You expect XCOPY to search in all the subdirectories and find matching files to copy to another subdirectory. That is asking too much of the command. Instead of obeying your order, DOS will give you this error message:

```
Cannot perform a cyclic copy
     0 File(s) copied
```

The reason for the refusal is that the target directory, namely C:\DOS, happens to be one of the source directories. Without this inhibition, files would be copied onto themselves.

The /S switch alone will not transfer an empty subdirectory in the source to the target. If you want an empty subdirectory found in the source also duplicated in the target, you need to combine /S with /E, such as:

```
A>xcopy *.* b: /s/e
```

This command will transfer everything from drive A to B. All the files and subdirectories will be transferred. Without the /E switch, only subdirectories containing files will be created in drive B. The empty subdirectories, if any, will be ignored.

Using the /S switch is convenient. It saves you the trouble of having to create target directories first before doing any copying to individual directories. However, if you use it indiscriminately, you could give yourself more trouble. If you are not happy with an existing directory structure and you use /S, you will just duplicate the messy arrangement, thus requiring you to spend more time to clean it up.

Prompt, Wait, and Backup

When you use the /P switch, DOS will show each file to be copied and prompt you for Y or N. This is most useful if wild card characters are used. It lets you decide which of the matching files to copy, as shown below:

```
A>xcopy chap* b: /p
CHAP1 (Y/N)? _
```

If you press N, this file is not copied. If you press Y, it is copied. The next matching file will be displayed and you will be prompted until the last matching file is acted on.

The /W switch instructs DOS to wait before copying. This message appears before any action, reading or writing, is done:

```
A>xcopy file??.txt b: /w
Press any key to begin copying file(s) _
```

This is useful if you want to use a drive to load the XCOPY.EXE program and then use it as a source or target drive. After the program is loaded, you can take the disk out and use the drive for copying.

The /A, /D, and /M switches are all related to backing up files. They let you copy only the files marked with the archive bit (/A), created on or after a certain date (/D), and remove the archive attribute from the source files that have been copied (/M). These processes will be explained in greater detail when we discuss file backup in the next chapter.

XCOPY's Limitations

XCOPY does not recognize the DOS device names AUX, COM1, CON, NUL, and PRN (or LPT1). If you try to copy a file to one of these devices, you will get the following response:[3]

```
A>xcopy chap1.txt prn
Does PRN specify a file name
or directory name on the target
(F=file,  D=directory)? f

Cannot XCOPY to a reserved device
       0 File(s) copied
```

If you answer D in the above example, you will get this error message:

```
Unable to create directory
       0 File(s) copied
```

Another point to remember is that XCOPY, like COPY, cannot copy to an unformatted disk. Unlike DISKCOPY, XCOPY cannot format a disk.

Finally, do not combine files with XCOPY. You will either get the Invalid parameter message or the result will surprise you. This is an example:

```
A>xcopy file1 + file2
```

This command will be interpreted as copying the first file to the second, not appending the second file to the first as in the case of COPY.

[3] You probably remember that if you use COPY in this situation, the named file will be printed without further ado.

Practice 6-3

☐ 1. Place the disk containing the previously created test files in drive A. Create a subdirectory with the name of `TESTDIR`. Copy `TEST1` to this directory and delete it in the root directory.

☐ 2. Place another practice disk in drive B. Use `XCOPY` to copy all the test files from A to B. Make sure you use three switches: one to make DOS wait for your move before copying, one to prompt you for `Y/N` before copying a file, and one to copy subdirectories.

☐ 3. Go to drive B and examine the results of the above command.

Drill

_____ 13. Which command is illegal?

a. `A>copy con prn`
b. `A>copy filename prn`
c. `A>copy con filename`
d. `A>xcopy filename con`

T 14. When `XCOPY` is used with the `/W` switch, copying begins immediately. True or false?

F 15. When `XCOPY` is used with `/E` alone, empty directories in the source drive will be copied to the target. True or false?

T 16. `XCOPY`'s `/S` switch will copy only the directories that contain files, but not empty ones. True or false?

F 17. `XCOPY` can be used to combine files. True or false?

T 18. A target disk must be formatted when you use `XCOPY`. True or false?

Copying a Whole Disk with `DISKCOPY`

✎ **Use `DISKCOPY` to duplicate a disk track by track.**

You first encountered the `DISKCOPY` command in Chapter 2. Here you learn more about its details.

This external command can be used only with floppies, not hard disks. It copies from source disk to target disk track by track, not file by file as in the case of `COPY` and `XCOPY`.

When you run the command, it is loaded to the memory and resides there until you terminate it. This frees the drives for copying files. To copy from drive A to B, put the disk containing the DISKCOPY.COM program in drive A and enter the following command, which leads to the instructions below:

```
A>diskcopy a: b:

Insert SOURCE diskette in drive A:

Insert TARGET diskette in drive B:

Press any key to continue . . .

Copying 40 tracks
9 Sectors/Track, 2 Side(s)

Volume Serial Number is xxxx-xxxx

Copy another diskette (Y/N) ?n

A>_
```

After the command is entered and the program loaded, you can take out the disk and replace it with the source disk to copy to the target.

Make sure you do not mix up the source and the target disks. If you reverse them, you could wipe out the original and replace it with garbage. One way to protect yourself is to place a write-protect tab over the source disk. This will prevent it from being overwritten with something else.

If you have one floppy drive, you can enter the above command or omit the drive specifications. After DOS reads the source disk, it will prompt you to insert the target disk. If there is not enough memory for a one-pass operation, you may be prompted to swap disks several times to copy just one disk.

The DISKCOPY command will wipe out the target disk's contents. If necessary, it will format the target disk according to the specifications of the source disk. The target disk's contents, if any, will be erased before copying.

DISKCOPY will duplicate a bootable disk. If the source disk is bootable, i.e., formatted with FORMAT/S, the target will also be bootable. This command is most useful when you need to duplicate an original system disk.

If you want to copy only one side, use the /1 switch, such as:

```
A>diskcopy a: b: /1
```

This is the only switch DISKCOPY has.

Comparing COPY, XCOPY, and DISKCOPY

- **Differences and similarities among the commands are explained.**
- **Use COPY or XCOPY, but not DISKCOPY, to duplicate a fragmented disk.**

In this section we compare COPY, XCOPY, and DISKCOPY.

COPY copies one file at a time; XCOPY reads source files to the available memory before writing to the target. This difference makes XCOPY work faster, particularly if you have ample RAM and need to copy large files.

COPY can copy files to devices, combine them, change their date and time, and read or write files in ASCII format. XCOPY can do none of these things.

XCOPY can copy subdirectories, prompt you for Y/N before copying a matching file, wait for you to insert a source or target disk, and copy specially marked files. COPY cannot do these.

DISKCOPY can copy to an unformatted disk and will erase the contents of the target disk, but COPY and XCOPY require the target disk to be formatted and will only add to the target disk's contents.

Both COPY and XCOPY copy each file as a unit. DISKCOPY physically duplicates a whole disk's contents. If a disk is used often, files will be fragmented. One file may be stored in different noncontiguous sectors. This could slow down access time and prematurely wear out the disk drive mechanism. If you use DISKCOPY, such a disk will be duplicated exactly, fragmentation and all. COPY or XCOPY, however, can cure this problem.

If a disk is used often for writing and deleting files, it is a good idea to periodically go through a simple exercise to reintegrate each file (put together all the scattered pieces). Put the used disk in drive A and a formatted blank disk in drive B and enter either command below:

```
A>copy *.* b:   (or A>copy . b:)
A>xcopy /w *.* b:   (or A>xcopy /w . b:)
```

In the first example, copying begins right away. In the second example, XCOPY will be loaded and wait for you to press a key; you can then replace the disk in drive A with a fragmented source disk.

Either command will put each file together and transfer it to the new disk. After that, each file will stick together physically and occupy contiguous sectors—until more writing and deleting cause new fragmentation.[4]

Copying Copy-Protected Software

✎ **DOS commands cannot duplicate copy-protected disks.**

During the early days of the PC revolution, "pirating" (making illegal copies of software) was common. Individuals got illegal copies of expensive packages from their friends or coworkers without paying for them. Vendors responded by adding devices to prevent copying. Legitimate users vehemently complained about the inconvenience they had to go through. They could not install on their hard disks

[4] To defragment a hard disk is more complicated and you need special software to do it. There are quite a few available on the market today. The popular ones include Disk Optimizer, Mace Utilities, Norton Utilities, and PC Tools. You can also use them to unerase (restore) the files you may have inadvertently erased on a hard or floppy disk.

more than a few times (mostly limited to five). Every time they wanted to use a package, they had to put the master disk in drive A. If the master disk where the protective device was added was ruined, they had to pay an extra fee and suffer considerable delay to get a replacement.

In response, vendors dropped copy protection and instead filed lawsuits against big companies, where illegal copying was rampant, to recover lost revenues. Today, most software packages sold in this country are no longer copy protected.

If you have a package that is copy protected, you cannot use COPY, XCOPY, or DISKCOPY to copy it. Either the copying will not succeed or the target copy will not run as expected.

To counter copy protection, some vendors came up with anti-anti-copying software. The most popular are COPYIIPC and COPYWRITE. If you use copy-protected software, you need one of these packages to make backup copies. As a legitimate owner, you have the right to make backup copies for your own use.

These packages are very easy to use. In the case of COPYWRITE, just enter COPYWRIT and the screen will display all the options, including source and target drives. With COPYIIPC, you just insert the software in drive A and issue this command:

```
A>copyiipc a: b:
```

You will then be instructed to put the source disk in drive A and the target disk in drive B. If you have a single floppy drive, you will be asked to swap the source and target disks, thus taking more time to complete the copying.

Comparing Files with COMP

✎ **Compare two individual files or two sets of files with this PC-DOS command. Differences appear in offset addresses and hex numbers.**

After making copies of existing files, you can use an external command named COMP (for PC-DOS) to compare them. Its works like this:

```
A>comp file1 file2
```

Here you ask DOS to run the COMP . COM program from the default drive and compare FILE1 and FILE2. If both files are entered, DOS will immediately go to work and then show the results. If one file is missing, you will be asked to enter it.

If no file follows the command, DOS will prompt you to enter two files to compare and then show the results, as in the following dialogue:

```
A>comp

Enter primary file name
file1                                   (entered by you)

Enter 2nd file name or drive id
b:file1.bak                             (entered by you)

A:FILE1 and B:FILE1.BAK
```

```
EOF mark not found
Files compare OK
Compare more files (Y/N)? n

A>_
```

DOS notices that the files do not have ^Z characters at the end. These are files created with COPY CON without using the /A switch. The absence of this character does not prevent a successful comparison. When the Files compare OK message appears, you can rest assured that the two files are identical.

If the files to be compared are of different sizes, you will get this message:

```
Files are different sizes
```

No further analysis is performed and the comparison is terminated. This could happen even if you have two identical files but with one byte difference. This one byte difference can result from using the /A switch to copy a file, as explained earlier. The target file will have an extra ^Z character added to the end, and the file size will be one byte longer than the original.

If the files are identical, analysis will proceed and errors, if any, will be displayed. However, you need a little computer knowledge to decipher what DOS tells you.

Let's create two files and then compare them. Follow these steps:

1. Create the first file:

```
A>copy con file1
This is a file.
^Z
```

2. Create the second file:

```
A>copy con file2
This is 1 file.
^Z
```

3. Compare the two files:

```
A>comp file1 file2
```

The results are shown below:

```
Comparing A:FILE1 and A:FILE2
Compare error at OFFSET 8
File 1 = 61

File 2 = 31

Eof mark not found

Compare more files (Y/N)? n
```

The message tells you that there is one difference in OFFSET 8, which is the ninth position (counting 0 as the first). DOS detects the value there as 61 in File 1 and 31 in File 2. Hex 61 (hexadecimal 61 = 6 x 16 + 1 = 97 decimal) is the value of a and hex 31 (49 decimal) is 1. (Appendix B shows you the characters and their

corresponding decimal values. Hexadecimal is a base 16 system, in contrast to the base 10 decimal system. See Appendix F for details.)

You can use wild cards to compare two sets of files, such as:

```
A>comp *.txt b:*.bak
```

If many errors are found, the message display will quickly scroll off the screen. One remedy is to send it to the printer for further review, by using this command:

```
A>comp *.txt b:*.bak > prn
```

If the printer is turned on, the message will be sent to the printer instead of the screen.

Comparing Files with FC

✎ **Compare two individual files (no wild cards allowed) with this MS-DOS command. Differences are displayed as ASCII text on the screen.**

Another command you can use to compare files is the external command called FC (FC.EXE); it is available in MS-DOS, but not PC-DOS. This command takes a different approach in comparing files. If the two files are identical, you get this message:

```
A>fc file1 file2
fc: no differences encountered
```

If there is a difference, you get a much less cryptic message like this:

```
A>fc file1 file2
***** file1
This is a line.
***** file2
This is 1 line.
*****

A>_
```

Instead of the offset numbers and ASCII values displayed by COMP, you get the comparable sets of the two files displayed for your view. Even if the two files are different sizes, comparison will proceed.

FC runs only once and is limited to two specific files. When a comparison is finished, you are not prompted for another. You cannot use wild card characters with FC. DOS will refuse to open a file for comparison.

If you have two large files to compare, you may want to display an abbreviated version by using the /A switch:

```
A>fc /a file1 file2
```

This displays only the beginning and ending lines for each set of differences. The screen is not cluttered with the whole text.

If you want the comparison to ignore cases and treat uppercase and lowercase the same, you can use the /C switch:

```
A>fc /c file1 file2
```

When binary files are encountered, those with COM, EXE, SYS, OBJ, LIB, and BIN extensions, FC uses binary mode as the default, which will result in a display of address locations and mismatched bytes, numbers that are more confusing than what you see in using COMP. When these are not encountered, ASCII is the default. You can change the default by using the /B (binary) or /L (ASCII) switch.

When no file name is entered after FC, this message appears:

```
A>fc
usage: fc [/a] [/b] [/c] [/l] [/lbNN] [/w] [/t] [/n] [/NNNN] file1 file2
```

So far we have covered the most commonly used switches. See Appendix D for explanations of the others.

Comparing Disks with DISKCOMP

✎ **Compare two floppy disks (not hard disks) with DISKCOMP. Comparison is meaningful only between an original and a duplicate created with DISKCOPY.**

When you have two diskettes you want to compare, you can use DISKCOMP, which is an external command. You need to enter drive specifications after the command. If not, this message appears:

```
A>diskcomp
Invalid drive specification
Specified drive does not exist
or is non-removable
```

From the message, you know that you must enter drive specifications and can compare only diskettes, not a hard disk. If you enter two floppy disk drives, the comparison will proceed as shown below:

```
A>diskcomp a: b:

Insert FIRST diskette in drive A:

Insert SECOND diskette in drive B:

Press any key to continue . . .

Comparing 40 tracks
9 sectors per track, 2 side(s)

Compare OK

Compare another diskette (Y/N) ?n
A>
```

Can you compare two diskettes even if you have only one floppy drive? Yes. Just issue the above command. You will be prompted to insert the first diskette, then the second. The comparison will then begin.

Hard Disk Users: If you have a hard disk and store your DOS programs in the directory named DOS, you can use either command below:

```
C>\dos\diskcomp a: b:
C>\dos\diskcomp a: a:
```

If you have two floppy drives, use the first. In case of one drive, either command will do; you will be asked to swap disks. If either drive specification is missing, DOS will interpret your intent as comparing a floppy and a hard disk, and refuse to oblige.

If comparison results in a discrepancy, errors are reported like this:

```
Compare error on
side 0, track 15

Compare error on
side 1, track 15

Compare error on
side 0, track 1^C
```

To end further comparison, press Ctrl-C. If errors are found, it makes no sense to continue the comparison.

`DISKCOMP` compares two disks track by track. It is meaningful to compare an original and one produced by `DISKCOPY`. If `COPY` or `XCOPY` is used, the target's contents may be rearranged and not identical to the source. `DISKCOMP` may thus find discrepancies between such disks.

Practice 6-4

- ☐ 1. Compare the two practice disks; cancel the comparison if an error is found.
- ☐ 2. Use `COMP` (or `FC` if you use MS-D0S) to compare the three test files created earlier. The results should be redirected to the printer. Use wild cards where appropriate.
- ☐ 3. Use wild cards to delete the test files on both disks, and also remove the directories created earlier.

Drill

T 19. A target disk does not have to be formatted when `DISKCOPY` is used. True or false?

T 20. DISKCOPY duplicates tracks, but COPY and XCOPY duplicate files. True or false?

F 21. XCOPY can duplicate a bootable disk. True or false?

T 22. DISKCOPY will not defragment a fragmented disk as COPY or XCOPY will. True or false?

D 23. Which of the following is legal?

a. `A>diskcomp a: c:`
b. `A>diskcopy c: a:`
c. `A>fc file??.txt b:file??.bak`
d. `A>comp file*.txt b:file*.bak`

For questions 24–30: Answer E for external command, or I for internal command.

I 24. COPY

E 25. XCOPY

E 26. COMP

I 27. TYPE

E 28. DISKCOPY

E 29. DISKCOMP

I 30. VERIFY

Summary and Review

Check each item as you review and understand the following highlights of the chapter.

- ☐ 1. Use the COPY command to create duplicate files with the same names or new names. Use wild card characters to speed up action.
- ☐ 2. Use COPY to append files; place the target file at the beginning, followed by a plus (+) and each source file.
- ☐ 3. Use COPY to combine files; connect source files with a + (or wild cards), and place the target at the end, separated from the rest by a space.
- ☐ 4. Use COPY to place new date and time data on a file by putting a + at the end of the target. Make sure to use the /B switch if a binary file is involved.
- ☐ 5. Use the /V switch with COPY to verify the target file. Use /A or /B to read or write to ASCII or binary files.
- ☐ 6. Use COPY to copy files to devices such as CON and PRN. Files can be combined and sent to these devices.

- ☐ 7. Use XCOPY to speed up copying large files. Use the /S switch to copy subdirectories, /S/E to copy even empty directories, /P to display a Y/N prompt, /W to wait for your action.
- ☐ 8. Copy an original disk, particularly the system disk, with DISKCOPY. However, do not use this command to copy a fragmented disk.
- ☐ 9. Use COPY *.* or XCOPY *.* to reintegrate the fragmented files. File fragmentation occurs when a disk is used repeatedly for saving altered files.
- ☐ 10. If you have PC-DOS, compare files with COMP. Errors are shown as addresses and ASCII values in hex numbers.
- ☐ 11. If you have MS-DOS, use FC rather than COMP to compare files. The comparable text lines are displayed for your review. They are easier to understand than the error messages of COMP.
- ☐ 12. Compare diskettes, not hard disks, with DISKCOMP. If there are errors between the original supplied by the vendor and your copy, use DISKCOPY to make another copy of the original.

Review Questions

Write an answer for each question below.

1. What does the following command do?

```
A>copy b:*.doc *.bak
```

2. What does this command do?

```
A>copy filename.ext b: +
```

3. What does this command do?

```
A>copy filename.ext + b:
```

4. What does the following command do, and will the target file size be the same as the sum of all the source files?

```
A>copy *.ext all.ext
```

5. What is the default mode of COPY when simple copying is done? What about when files are combined?
6. How can you stamp the current date and time on a single file?
7. What differences do the following commands produce?

```
A>copy *.* b:
A>diskcopy a: b:
```

8. What does this command do?

```
A>copy filename.ext b: /a
```

9. Give a command that will copy a disk file named FILENAME to the printer.

10. What is the difference between the following two commands?

```
A>type filename
A>copy filename /b con
```

11. What does XCOPY's /S switch do?

12. What does this command do?

```
A>xcopy *.* b: /s/e
```

13. Can you use XCOPY to combine files or copy them to a device?

14. How do COMP and FC differ?

15. Do COMP and FC accept wild card characters in file names?

Version Comparison Chart

Commands/Features	PC-DOS	MS-DOS
Compare files	COMP	FC

Maintaining Files

Chapter 7

TOPICS:

Using a New DOS Version
Replacing Old Files
Setting and Removing File Attributes
Backing Up Files with XCOPY
Backing Up Files with BACKUP
Restoring Backup Files
Printing in the Background

COMMANDS COVERED:

SYS, REPLACE, ATTRIB, XCOPY, BACKUP, RESTORE, PRINT

NEW TERMS AND SYMBOLS:

archive, read-only, computer virus, exit code, print queue

REQUIRED MATERIALS:

1. Two practice disks
2. Access to the following files:
   SYS.COM, REPLACE.EXE, ATTRIB.EXE, XCOPY.EXE, BACKUP.COM,
   RESTORE.COM, PRINT.COM

Quick Start

✎ **To transfer the two hidden system files to the hard disk:**

```
A>sys c:
```

✎ **To replace files (replace target with source):**

```
A>replace source b:target
```

✎ **To show file attributes:**

```
A>attrib file?
```

✎ **To add attributes (to remove, replace + with −):**

```
A>attrib +a +r file?
```

✎ **To use XCOPY to copy a new file and remove its A attribute:**

```
A>xcopy filename b: /a/m
```

✎ **To back up all matching files (create a new backup file):**

```
C>backup file? a:
```

✎ **To back up only modified files (create a new backup file):**

```
C>backup file? a: /m
```

✎ **To add modified files to an existing backup file:**

```
C>backup file? a: /a/m
```

✎ **To restore backup files (must have a space after a:):**

```
C>restore a: file?
```

✎ **To send files to the print queue:**

```
A>print file?
```

After using your computer for an extended period of time, you are likely to accumulate a large number of files and possibly a large number of large files. You may need to back up these files for safety. You may have to selectively replace or restore them in various storage media.

Backing up files is vital. Electronic media such as hard or floppy disks are fragile. The data stored on them could be wiped out in a flash. If you have no backup copies, you will lose your work and possibly your entire business. In the computer press, there are continuous horror stories of individuals and companies suffering immeasurable losses due to data loss and a lack of backup files.

In case of emergency, you can use commercial utility programs such as Norton Utilities and PC Tools to restore corrupted or even erased files. Some of these utilities may recover most of your lost data as long as you have not saved over your files or reformatted the disk.

A more cautious approach is to back up your files religiously. If you have small files or a small number of files, use ordinary diskettes to make duplicate or triplicate copies. If one is corrupted, you can resort to another. For these files, commands such as `COPY`, `XCOPY`, and `DISKCOPY` will handle such jobs adequately.

If you have large files, particularly a large number of them, you need to use the more uncommon commands discussed in this chapter. They include `BACKUP`, `RESTORE`, and `REPLACE`.

The ultimate security you can give to a file is to make a hard copy. Should everything go wrong, you will still have the printout. To that end, you will learn in this chapter how to use `PRINT.COM` to do background printing.

Using a New DOS Version

✎ Use `SYS.COM` to transfer hidden system files to a boot disk.

When you get a new DOS version, you can use `DISKCOPY` to duplicate the system disk and boot your computer from it. Or you can format a disk with the `/S` switch to make it bootable. If you have a hard disk, you need to use the `SYS` command (`SYS.COM`) to transfer the system (hidden) files so that it can boot with the new version. Here are the steps to do it:

1. Insert the new system disk (labeled Install for version 4) in drive A and turn on the power (or press the Reset button or press Ctrl-Alt-Del if power is already on).

   If you use version 3.3 or earlier, you are asked to enter the date and time. Then the `A>` prompt appears. If you use these versions, go to step 2. If you use version 4, the following appears on the screen:

   ```
   Insert SELECT diskette in drive A:

   Press Enter to continue installing DOS,
   or press Esc to exit.
   ```

   Press Esc, and the `A>` prompt appears.

2. Use `DIR` to see whether `SYS.COM` is on the disk in drive A. If not, try another disk. (In version 4, the program is stored on the Install disk; this step can be omitted.)

3. Use `SYS.COM` to transfer the hidden system files:

   ```
   A>sys c:
   ```

   The drives (A and C) begin to make noise. After a while, this message appears:

   ```
   System transferred
   ```

4. Transfer COMMAND.COM:

```
A>copy command.com c:
```

After DOS tells you that one file has been copied, the transfer is complete. If you now leave drive A's door open and press Reset or Ctrl-Alt-Del, the new version will be booted.

Note: One common mistake is forgetting to copy COMMAND.COM after using SYS. SYS, unlike FORMAT, does not transfer this file to the target disk. If your hard disk will not boot after using SYS, the most likely cause is the absence of the new COMMAND.COM file.

Version Variances: In versions before 4, the hidden system files must be on the disk where SYS.COM is stored, and you can specify only a target drive but not a source drive. In version 4, you can specify a source drive where the system files are stored, such as:

```
A>sys c: b:
```

This will load SYS from drive A and transfer the system files from drive C to drive B. If you then copy COMMAND.COM to drive B, it will be bootable.

Tips & Tricks

Installing Versions 3 and 4

Version 3.3 or earlier requires the two hidden system files of IBMBIO.COM (IO.SYS for MS-DOS) and IBMDOS.COM (MSDOS.SYS) to be the first two entries on the boot disk and IBMBIO.COM to be stored in contiguous sectors (not divided). As newer version files are invariably larger than the older version, you cannot squeeze them into the same space and are likely to encounter this error message:

```
A>sys c:
No room for system on destination disk
```

If this happens, you may have to delete some vital files or reformat the entire disk.

Version 4 no longer imposes these requirements. You can thus use SYS to transfer the system files without any problem.

When you develop your own computer program, you can store it on a disk formatted with the /B switch. This provides the space for storing system files later. When you use the SYS command to transfer system files, the disk becomes bootable.

After you finish copying the hidden files and COMMAND.COM, you need to replace the old DOS external commands on your hard disk with the new ones. If

you boot with the new version, the old external commands will no longer work. Conversely, if you boot with the old version, you cannot run the new version's external commands. If you try either way, you will most likely get this error message:

```
Incorrect DOS version
```

You can use `COPY` or `XCOPY` to replace the old version's files. A more convenient way, however, is to use the `REPLACE` command. `REPLACE` can also be used to replace other files. It is discussed in the next section.

Tips & Tricks

Changing from PC-DOS to MS-DOS

If you replace PC-DOS with MS-DOS or vice versa, `SYS.COM` will copy the hidden system files from a floppy to your hard disk. However, since the hidden files of the two brands do not share the same names, the old hidden files will remain and take up your disk space. You cannot use `DEL` or `ERASE` to get rid of them.

One way out is to use Norton Utilities or similar products to remove the hide attribute and then delete them. Computer bulletin boards and computer magazines often make available to the public a simple assembly program that can change such an attribute. Look for a program with a name like `CHMODE.COM`, `HIDE.COM`, or `UNHIDE.COM`.

Drill

F 1. `SYS.COM` can be used to transfer both hidden and unhidden files. True or false?

T 2. DOS contains two hidden system files. True or false?

T 3. In versions 3.3 and before, if the hidden system files are not the first two directory entries on the boot disk, the disk will not boot. True or false?

F 4. If you boot from a new DOS version, you can still run most external commands of the old version. True or false?

Replacing Old Files

✎ **Use REPLACE to replace or update old DOS files or your own files.**

The REPLACE command can be used to replace old files with new ones. In its simplest form, it can behave like COPY. But when its various switches are used, it can do much more to maintain files.

The Syntax of REPLACE

REPLACE's syntax resembles COPY's:

```
A>replace source target
```

This will replace target files with source files, or place source files in the target.

Normally, when you think of the English verb "replace," you think of replacing the first object with the second object. The situation is reversed with REPLACE. If you think of replacing the second file with the first, you are closer to the syntax of REPLACE.

If there is only one parameter, it is presumed to be the source and the target is the default drive, such as:

```
A>replace b:file1
```

This will replace FILE1 in drive A with FILE1 in drive B.

The Switches of REPLACE

REPLACE has these switches:

/A	Add files that are in source but not in target
/P	Prompt for Y/N to replace old files
/R	Read-only files are also replaced
/S	Subdirectories in target; files there replaced
/U	Update old files with new (version 4)
/W	Wait for key press before acting

REPLACE is very handy in replacing the old DOS files on your hard disk with the new ones. After you boot up with the Install disk of version 4, you can run REPLACE to replace the old files with a command like this:

```
A>replace *.com c:\ /s
```

This will replace all the COM files in drive C from the root directory down to all the subdirectories with files of identical names found in drive A. This is a very sweeping command. It will automatically replace and display every matching file. If you want to be prompted for Y/N before each file is replaced, try this:

```
A>replace *.com c:\ /s/p
Replace C:\DOS\DISKCOPY.COM? (Y/N)_
```

If you press Y, the displayed file is replaced. If you press N, it is not replaced.

You can use the /A switch to add the files to the target drive that exists in the source but not in the target, such as:

```
A>replace *.exe c:\dos /a/p
```

This will add any file to C:\DOS that is found in drive A but not in C:\DOS. The old files in the target are not replaced.

Version Variances: Version 4 has a new switch, /U. You can use it to update (replace) older files with newer ones. There is no adding involved. Only files with identical names are compared and replaced. If two identical files have the same date-time, no replacing occurs. This avoids repeated replacing of identical files.

When the /A switch is used, you cannot use /U or /W. If you did, you would get this error message:

```
Invalid parameter combination
```

REPLACE ends after the specified files on the source disk are copied to the target. If you want to replace another disk, you have to run it again. Since the program is stored on only one disk, you can use the /W switch to load it and then wait for you to insert a new disk to copy, such as:

```
A>replace *.com c:\dos /w/a/p
Press any key to continue . . .
```

If you use version 4, you can now take out the Install disk and insert the Select or another disk to replace your old DOS files. If you use other versions, replace the first disk with the second.

The /R switch is used to overcome the R (read-only) attribute. When a target file is designated as read-only (explained below), you must use /R to replace it with a new one. If not, the remaining replacing operation will end when such a file is encountered; the screen also shows the file's name and the Access Denied message. If /R is used, all the matching target files will be replaced.

Replacing Your Own Files

REPLACE can also be used to replace the files you create. To demonstrate how to use it on your own files, we will create a few and then replace some. Before you do, make sure REPLACE.EXE is stored in drive A or accessible through the PATH command. Insert a practice disk in drive A and another in drive B, and follow these steps:

1. Create the first file:

```
A>copy con file1
This is file 1.
^Z
```

2. Add this file to drive B:

```
A>replace file? b: /a

Adding B:\FILE1

1 file(s) added
```

This adds to drive B all the matching files found in drive A but not in drive B.

3. Change the contents of the file in drive B:

```
A>copy con b:file1
This is a new file.
^Z
```

4. Update the old file with the Y/N prompt:

```
A>replace b:file? /u/p

Replace A:\FILE1? (Y/N)y
Replacing A:\FILE1

1 file(s) replaced
```

5. Show the file in drive A:

```
A>type file1
This is a new file.
```

The original FILE1 in drive A has been replaced with the one from drive B carrying the same name but with a later date-time.

Practice 7-1

- ☐ 1. Create and save in drive A two simple files named TEST1 and TEST2. Use REPLACE to copy the matching files (TEST?) to drive B.
- ☐ 2. Delete the two matching files in drive A. Use the /A, /P, and /W switches with REPLACE to copy all matching files from B to A.
- ☐ 3. Add a line to TEST1 in drive A. Use the /U switch with REPLACE to copy matching files to drive B.

Drill

T 5. If you insert a DOS disk in drive A and enter the command C:\>replace a:*.* /s, all the matching files in drive C (including all the subdirectories) will be replaced with the files of identical names in drive A. True or false?

F 6. The command C:\>replace a:*.* /a will replace old files and add new files from the source to the target. True or false?

A 7. The command A>replace project?.txt b: will:
   a. replace B's matching files with those from A
   b. replace A's matching files with those from B
   c. add files to B that are in A but not in B
   d. add files to B that are in B but not in A

b 8. The command `A>replace b:project?.txt /a` will:
   a. replace B's matching files with those from A
   b. replace A's matching files with those from B
   c. add files to B that are in A but not in B
   d. add files to A that are in B but not in A

F 9. If `PROJECT.TXT` exists in A but not in B, the command `B>replace a:project.txt /u` will copy it from A to B. True or false?

Setting and Removing File Attributes

✎ **Use ATTRIB to set or remove archive and read-only attributes.**

When you have many existing files, you can set certain attributes on them. These attributes can then be used to manipulate the files, for example, copying or replacing them.

As explained in Chapter 3, file attributes, if any, are stored in a directory. You can use the `ATTRIB` command to display them. You can also use it to set or remove these attributes. For example, to set a file to read-only, this command will do:

```
A>attrib +r filename
```

You can use four attributes and one switch with `ATTRIB`:

`+R`	Set read-only
`-R`	Remove read-only
`+A`	Set archive
`-A`	Remove archive
`/S`	Subdirectories

To demonstrate how to use `ATTRIB`, let us create a file and then set and remove its attributes. Before you do, make sure `ATTRIB.EXE` is stored in drive A or accessible through the `PATH` command.

1. Create `FILE1`:

```
A>copy con file1
This is file 1.
^Z
```

2. Show its attributes:

```
A>attrib file1
  A            A:\FILE1
```

The **archive** (`A`) attribute is automatically set as the default when a file is created or modified. This attribute can be used by `XCOPY`, `BACKUP`, and `RESTORE` to move files.

3. Set the read-only attribute:

```
A>attrib +r file1
```

4. Show the new attributes:

```
A>attrib file1
  A     R       A:\FILE1
```

5. Delete FILE1:

```
A>del file1
Access denied
```

6. Add a line to FILE1:

```
A>copy file1 + con
FILE1
Access denied  - A:FILE1.
        0 File(s) copied
```

When the **read-only** (R) attribute is set, a file can only be read; it cannot be added, deleted, or replaced. If you attempt any of these moves, you will be denied access.

7. Remove all attributes:

```
A>attrib -a -r file1
```

8. Show attributes:

```
A>attrib file1
                A:\FILE1
```

9. Set archive attribute:

```
A>attrib +a file1
```

This file has now been returned to its original state.

> **Note:** You can use wild card characters with ATTRIB or to process all the matching files in subdirectories, such as:
>
> ```
> A>attrib +a +r b:\file?.* /s
> ```
>
> Each matching file found in drive B's root directory and all the subdirectories, if any, will be processed. Your disk will make some noise, but the files being processed are not displayed.

Backing Up Files with XCOPY

✎ **Use XCOPY to copy archived or dated files.**

Among the commands that can be used to back up files, XCOPY looks most familiar. We have covered some of its switches in the previous chapter. We are going to use the rest for backing up files.

As you have learned in Chapter 6, XCOPY contains, among others, these switches:

/A	Archive; copy files marked with A
/D:date	Date; copy files created on or after date
/M	Remove source archive bit after backup

Tips & Tricks

Protection against Viruses

A few days before Columbus Day in 1989, the media carried headline news about a **computer virus** poised for attack nationwide. It was said to start attacking on Columbus Day; that is why it was referred to as the Columbus Day virus. Its targets were certain files that carry the COM extension.

Computer security experts who were interviewed offered wide-ranging advice such as insulating the system from unsanitized bulletin boards, refusing to use programs from unproven sources, and protecting files with antivirus programs. The most useful and certainly most economical advice was to make vulnerable files read-only. This Columbus Day virus, like others, was said to be unable to attach itself to files having the R attribute.

We are going to demonstrate how to use them for backing up files. Before you do, make sure ATTRIB.EXE, XCOPY.EXE, and REPLACE.EXE are in drive A or accessible somewhere.

1. Set A and R attributes to FILE1, which was created earlier:

```
A>attrib +a +r file1
```

2. Copy files by the date:

```
A>xcopy file? b: /d:06-25-90
```

Specify the current date or earlier after the /D: switch; use the mm-dd-yy format. All the matching files on or after the specified date will be copied to the target disk.

3. Copy and remove the archive attribute:

```
A>xcopy file? b: /m
```

After this operation, the source files no longer carry the A attribute. The target files, however, still have it. This arrangement can avoid duplicating backup. If you use the /A or /M switch in the future, these files will not be copied because of the lack of the archive bit. However, you can still use XCOPY (or COPY) without any switch to copy such files.

4. Show source file attributes:

```
A>attrib file?
        R       A:\FILE1
```

Notice that the A attribute is gone, removed by the /M switch.

5. Copy archived files:

```
A>xcopy file? b: /a
        0 File(s) copied
```

Since the files have the A attribute removed, XCOPY cannot find any matching file to copy.

6. Show target file attributes:

```
A>attrib b:file?
  A          A:\FILE1
```

The A is automatically added regardless of whether a source file has it or not. Notice that the R attribute is not copied. This is true regardless of whether you use COPY, XCOPY, or REPLACE.

7. Copy matching archived files from B to A:

```
A>xcopy b:file? /a
Access denied  - A:\FILE1.
        0 File(s) copied
```

The original file in drive A carries the R attribute and thus cannot be overwritten.

8. Replace FILE1 in drive A with the same file in drive B:

```
A>replace b:file1
Replacing A\FILE1
Access denied  - A:\FILE1

No files replaced
```

9. Replace with the /R (read-only) switch:

```
A>replace b:file1 /r

Replacing A:\FILE1

1 file(s) replaced
```

This time the replacing is successful. The file in drive B has replaced the file in drive A.

10. Show the new file's attributes:

```
A>attrib file1
  A    R      A:\FILE1
```

The old file's attributes are not changed, only the contents are.

XCOPY, like COPY, stops copying when the target disk runs out of space. This could become a problem if you have a long file. The remedy to this situation is the BACKUP command explained below.

Tips & Tricks

Exit Codes

`REPLACE`, `XCOPY`, `BACKUP`, `RESTORE` and some other commands report an **exit code** in response to a particular state of execution. If a job is successfully completed, for example, each will report the exit code of 0. If an error of some sort is encountered, that particular exit code number is reported.

Exit codes can be used by the `IF` command in a batch file to test the state of a command's execution. If a job is successfully completed, the batch file can tell the user that. If a specific error is encountered, the batch file can warn the user with a specific message. This will be covered in Chapter 11.

Practice 7-2

- ☐ 1. Show the file attributes of `TEST1` and `TEST2` in drive A, created in the previous session.
- ☐ 2. Add the read-only attribute to the above files.
- ☐ 3. Use `REPLACE` to replace the two matching files in A with those from B.
- ☐ 4. Use `XCOPY` to copy the two files in A to B and at the same time remove the archive bit of each source file.
- ☐ 5. Delete the two files in A. Use `REPLACE` to add the two files in B to A. (These two files will be used in the next session.)

Drill

F 10. You can delete a file when the read-only attribute is set. True or false?

F 11. You can set or remove a file's attribute only one file at a time and one attribute each time. True or false?

F 12. After the first command, both the second and the third commands below will succeed in replacing `FILE1` in drive A with `FILE1` from drive B. True or false?

```
A>attrib +r file1
A>replace b:file1 /r
A>xcopy b:file1
```

T 13. Assuming `FILE1` also exists in B and is not set to read-only, the second command below will succeed in replacing that file. True or false?

```
A>attrib +r file1
A>replace file1 b:
```

F 14. If you use the `/M` switch with `XCOPY` to copy a file, you cannot use `XCOPY` to copy the file again without doing something to the file first. True or false?

C 15. The command `A>xcopy file? b: /a` will:
- a. add files to B that are found in A but not in B
- b. add files to A that are found in B but not in A
- c. copy only files with the archive bit
- d. remove source files' archive bit

B 16. The target file after the second command below will have:

```
A>attrib +a +r file1
A>xcopy file1 b:
```

- a. the same attributes as the source
- b. no `R` attribute
- c. no `A` attribute
- d. neither `A` nor `R` attribute

C 17. The second command below will:

```
A>attrib +r file1
A>replace b:file1
```

- a. replace the file in A with the one in B
- b. replace the file in B with the one in A
- c. display the `Access denied` message
- d. do nothing

F 18. If the target disk runs out of room, `XCOPY` will prompt you to insert a new disk and continue copying. True or false?

Backing Up Files with `BACKUP`

✎ **Use `BACKUP` to create sequentially numbered backup files that contain source files having certain attributes.**

✎ **`BACKUP` can handle large files. If more disks are needed, you will be instructed to insert them.**

If you need to do a big backup job requiring multiple disks, you need to use `BACKUP`. You will be prompted to insert a new disk each time one is full. The backed up files can be restored with `RESTORE` in the exact order of their backup. Number each backup disk so that they can be restored in the correct order.

BACKUP's Switches

BACKUP provides quite a few switches, as shown below:

/A	Add to files already on the backup disk
/D:date	Date on or after which to back up
/F:size	Format target by specified size (v.4)
/L:name	Log file name containing backup history
/M	Modified files only; changed since last backup
/S	Subdirectories are backed up
/T:time	Time at or after which to back up

These switches will be explained and illustrated in the following examples.

To demonstrate how to use BACKUP, we will create more files and then use the command's various switches.

BACKUP Requirements

Before you proceed, you need to store BACKUP.COM and FORMAT.COM in drive A. While you are at it, also copy RESTORE.COM to drive A; this command will be used in the next section. If you store the above commands in the root directory of drive A, enter this before proceeding further:

```
A>path=a:\
```

Hard Disk Users: If you have a different arrangement, such as storing various DOS external commands on a hard disk, use PATH to tell DOS where they are:

```
A>path=c:\dos
```

This tells DOS to go to the C:\DOS directory to look for an external command when one is issued.

To store backup files, you need to have in drive B (target) a formatted or unformatted disk. If it is unformatted, it will be formatted before backing up. If it is formatted, any file stored in its root directory will be erased before backing up; if there are files you intend to use later, you should transfer them before proceeding.

BACKUP Steps

With a disk in A and one in B, you are ready to go:

1. Create a subdirectory (covered in Chapter 5):

   ```
   A>md back
   ```

2. Make the new subdirectory current:

   ```
   A>cd back
   ```

3. Show the default drive and current directory:

   ```
   A>prompt $p$g
   A:\BACK>_
   ```

4. Move FILE1 to the new subdirectory:

   ```
   A:\BACK>copy \file1
   A:\BACK>del \file1
   ```

5. Create two additional files:

```
A:\BACK>copy con file2
This is the first line of file 2.
^Z

A:\BACK>copy con file3
File 3 is here.
^Z
```

6. Back up matching files in A to B:

```
A:\BACK>backup file? b:
```

You should now see the message below:

```
Insert backup diskette 01 in drive B:

WARNING! Files in the target drive
B:\ root directory will be erased
Press any key to continue . . .
```

If you are not sure whether you should go ahead, press Ctrl-C and then examine the contents of the target disk; if there are files you want to keep, transfer them to another disk or use another disk in drive B. Otherwise, press a key to start. The message below appears, and each file is displayed as it is backed up:

```
*** Backing up files to drive B: ***
Diskette Number: 01

\BACK\FILE1
\BACK\FILE2
\BACK\FILE3
```

7. Display the target drive:

```
A:\BACK>dir b:

 Volume in drive B is BACKUP  001
 Volume Serial Number is 0F70-1BFF
 Directory of  B:\

BACKUP   001         69 06-27-90   3:32p
CONTROL  001        311 06-27-90   3:32p
        2 File(s)     360448 bytes free
```

Note: Notice the volume label, which is `BACKUP.001`. This is the first backup disk. Other backup disks, if any, will be numbered sequentially. If a disk is full before a backup job is complete, you will be prompted to insert a new disk.

Notice also that there are only two files on the target disk, `BACKUP.001` and `CONTROL.001`. The former contains the contents of all the original files, and the latter stores the information (including file names and directories) for restoring these files.

8. Display the source files:

```
A:\BACK>dir file?

 Volume in drive A is DISK1
 Volume Serial Number is 1C10-5A08
 Directory of  A:\BACK

FILE1                17 06-27-90   3:19p
FILE2                35 06-27-90   3:19p
FILE3                17 06-27-90   3:19p
        3 File(s)          4096 bytes free
```

Notice that the source files add up to the same number of bytes as the BACKUP.001 file, namely 69 bytes.

9. Display the contents of the backup file:

```
A:\BACK>type b:backup.001

This is file 1.
This is the first line of file 2.
File 3 is here.
```

All the text lines of the source files are combined. If you try the same thing with CONTROL.001, you will get mostly gibberish plus a few readable characters.

Using BACKUP's Switches

We are going to apply BACKUP's common switches to do backup. Before we do that, we will first modify an old file and create a new file.

1. Add one line to FILE2 (created in step 5 above):

```
A:\BACK>copy file2 + con
FILE2
CON
This line is added to file 2.
^Z
```

2. Create a new file:

```
A:\BACK>copy con file4
File 4 is newly created.
^Z
```

3. Go to the root directory:

```
A:BACK>cd\
A:\>_
```

4. Back up with the following switches:

```
A:\>backup file? a: /a/m/s/l:backlog

Insert last backup diskette in drive B:
Press any key to continue . . .

*** Backing up files to drive B: ***
Diskette Number: 01

Logging to file A:\BACKLOG
\BACK\FILE2
\BACK\FILE4
```

This time we want to add (`/A`) all the modified (`/M`) files in the current directory and all its subdirectories (`/S`). We also want to keep a log (`/L:filename`) of this particular backup. The screen shows two files backed up. The first is modified due to an additional line. The second is a file that does not yet exist in the backup file.

5. Show the log file's directory information:

```
A:\>dir backlog

. . .

BACKLOG                   55 06-27-90    4:18p

. . .
```

This file is automatically saved to the root directory of the source drive. At this point it has 55 bytes and contains the characters shown below.

6. Show the log file's contents:

```
A:\>type backlog


6-27-1990  16:18:36
001  \BACK\FILE2
001  \BACK\FILE4
```

The date and time data are kept in this file. The backup disk number (`001`) and the names (and their directory path) of the two copied files also show up here.

The backup log file contains the history of the backup. In the future, when you use the `/L` switch and the same log file name, similar data (date, time, backup disk number, directory path, and file names) will be added to it. This file is for you to examine what you have done and serves no other purpose.

You do not have to enter a file name after the `/L` switch. In that case, DOS will create or append to a file named `BACKUP.LOG`.

7. Show the contents of the new backup file:

```
A:\>copy /b b:backup.001 con

This is file 1.
This is the first line of file 2.
File 3 is here.
This is the first line of file 2.
This line is added to file 2.
File 4 is newly created.
```

The two source files are appended to the backup file. The /A switch is the reason for this.

Note: Notice that in step 7 we use the COPY/B instead of TYPE command. If you use the latter, the above display will not show the last line. The reason is that when a line is added to FILE2 in step 1, a ^Z (end-of-file marker) character is added by the COPY command. This character is then transferred to the middle of the backup file. This prevents the TYPE command from reading beyond it. COPY used with the /B (for binary) switch can read and display this character. See Chapter 6 for more details.

BACKUP operates in binary mode and will read and copy the ^Z character. The RESTORE command, as explained below, also works the same way. Consequently, both commands can be safely applied to binary files.

8. Delete the files that have been backed up:

```
A:\>del file?
```

Other Switches

We have used all the BACKUP switches except /D (date), /T (time), and /F (format). With the first two, you are to enter date and time as you do with the DATE and TIME commands. Only the files matching the numbers or later are backed up.

Version Variances: The /F switch is not necessary with version 4—it is a default setting. The target disk will be automatically formatted, if it is not already. This requires FORMAT.COM to be accessible whenever BACKUP is run. If it is not, no backup is done and the Cannot find FORMAT.COM message is displayed. Earlier versions will back up without accessing FORMAT.COM unless you specify the /F switch, which is needed only when you use unformatted disks.

The numbers that can be entered after /F follow the same rules governing the /F switch of FORMAT. See Chapter 3 for details.

File Attributes and Drives

BACKUP uses a file's A attribute to perform its various functions. Every time a file is backed up, its A attribute is removed. A file without this attribute can still be backed up if you do not use the /M switch. However, when you use the /M switch alone or in combination with others, BACKUP will only back up files with the A attribute. If no matching files have the A attribute, this error message appears:

```
WARNING! No files were found to back up
```

The R attribute does not interfere with a file's backup or its restoration. The files created by BACKUP (BACKUP.001 and CONTROL.001, but not BACKUP.LOG) have the R attribute added. That means you cannot erase them unless it is removed.

BACKUP works with certain rules in regard to your disk drives. If the backup target is a diskette, the backup files are stored in the root directory. If it is a hard disk (files cannot be backed up to the same hard disk), they go to a directory named \BACKUP. Furthermore, the BACKUP.LOG file (if the /L switch is used) is saved to the source drive's root directory.

Tips & Tricks

BACKUP's Hazards

Beware of the role played by the /A switch. Anytime this switch is not specified, this warning message appears:

```
WARNING! Files in the target drive
B:\ root directory will be erased
Press any key to continue . . .
```

When BACKUP encounters no switch or any switch except /A, it will create a new set of backup files, erasing the old ones if they exist.

If /A is used alone or in combination with others, BACKUP requires the previously created backup files to add to them. If they are not available, no addition is made.

Backing Up with a Batch File

If you work with important files on your hard disk, you may want to maintain a regular backup schedule, maybe daily, weekly, or monthly. In that case a batch file like the one below may be helpful:

```
C:\BAT>copy con autoback.bat
@echo off
echo Insert a blank disk in drive A
pause
path=c:\dos;c:\bat
backup c:\wp\work\file? a: < cr
^Z
```

The @echo off line is to hide batch execution. This will be explained fully in Chapter 11.

This batch file needs another file named CR to work:

```
C:\BAT>copy con cr

^Z
```

CR has a single blank line and nothing else. This has the same effect as pressing Enter when you are asked to press a key after running BACKUP. This redirection technique will be explained in greater detail in Chapter 8.

When you run AUTOBACK.BAT, it will pause and you will be instructed to insert a blank disk in drive A. After you press a key, backing up will begin immediately.

This batch file assumes that you store BACKUP.COM, FORMAT.COM, and other DOS program files in C:\DOS and that you want to back up FILE? in the C:\WP\WORK directory to drive A. All of these can be changed to suit your circumstances. If you want to back up an entire directory, you can just specify the directory name, such as:

```
backup c:\wp\work a: < cr
```

This will back up everything in the C:\WP\WORK directory.

If you want to add another directory plus all its subdirectories, just add another (second) line like below:

```
backup c:\wp\work a: < cr
backup c:\ss\budget\*.wk1 a: /s/a < cr
```

The /S switch will include subdirectories. The /A switch will add to, not replace, the backup files.

Practice 7-3

- ☐ 1. Use BACKUP with the /L switch and the default name to back up TEST1 and TEST2 in drive A to drive B. (Make sure the root directory in B does not contain any file you intend to save.)
- ☐ 2. Create a short file named TEST3. Add this file to the backup file with the /L switch.
- ☐ 3. Use TYPE to display the backup file.
- ☐ 4. Use TYPE to display the log file.

Drill

T 19. If the target disk runs out of room, BACKUP will prompt you to insert a new disk. True or false?

F 20. If a hard disk is a backup target, the files in its root directory will all be erased. True or false?

T 21. If a floppy disk is a backup target, the files in its root directory will all be erased. True or false?

warned

F 22. A backup file can contain only ASCII text. True or false?

A 23. The command `A>backup project? b:` will:
a. erase all existing files and create a new backup file
b. add to the existing backup file
c. create a log file
d. back up only modified files

D 24. The command `A>backup project? b: /m` will:
a. erase all existing files and create a new backup file
b. back up only modified files
c. add to the existing backup file
d. do both a and b

D 25. The command `A>backup project? b: /m/a` will:
a. erase all existing files and create a new backup file
b. back up only modified files
c. add to the existing backup file
d. do both b and c

Restoring Backup Files

✎ **If original files are lost, use `RESTORE` to restore them from backup files created by `BACKUP`.**

If your original files are corrupted or inadvertently erased, you can restore them from the backup files.

Restoring from backup files is easy. Just use `RESTORE` to transfer all or specific files from a backup disk to the original directory. If you have made the arrangement as explained at the beginning of the previous section, you are ready to go.

Steps to Restore Files

Now follow these steps to restore the backup files:

1. Place the original source disk in drive A and the backup disk in drive B—the same arrangement discussed in the previous section.

2. Make `A:\BACK` the current directory:

```
A:\>cd back
A:\BACK>_
```

3. Restore matching files from B to A:

```
A:\BACK>restore b: file?
Insert backup diskette 01 in drive B:
Press any key to continue . . .
```

4. Press a key to start. The following message appears:

```
*** Files were backed up 06-27-1990 ***

*** Restoring files from drive B: ***
Diskette: 01
\BACK\FILE1
\BACK\FILE2
\BACK\FILE3
\BACK\FILE2
\BACK\FILE4
```

The backup disk's number, the drive letter, and the latest backup date are displayed. Each file, together with its directory path, is shown as it is restored.

Notice that `FILE2` is restored twice. That is because two versions were backed up earlier. When we added a line and then used the `/A` and `/M` switches with `BACKUP`, the new version was added to the original backup file. In restoring files, both versions are copied and the later copy replaces and erases the earlier one.

RESTORE's Rules

In step 3 above, drive B is the source and A the target. We intend to copy the files matching the pattern of `FILE?` from drive B to the current directory of drive A.

Warning: Make sure a space exists after `B:`. If not, this error message will appear:

```
No target drive specified
```

You can also specify a directory path before a file name, such as:

```
A:\BACK>restore b: a:file?
```

Notice that we use the original file names, not `BACKUP.001` as found on the backup disk. If you want to restore everything, try this:

```
A:\BACK>restore b: *.*
```

This restores all the files from B to the `A:\BACK` directory. The target of `RESTORE` must be the original source directory when `BACKUP` was run. If you use any other directory, you will get this error message:

```
WARNING! No files were found to restore
```

For example, if you are in the root directory of drive A, the command below will generate the error message shown above:

```
A:\>restore b: *.*
```

On the other hand, this command will work:

```
A:\>restore b: a:\back\*.*
```

This guides DOS to restore everything to the original source directory in drive A from which files were originally backed up.

If you want to restore a specific file instead of all the matching ones, all you need to do is to enter the file name, such as:

```
A:\BACK>restore b: file2
```

This will restore only FILE2, but not the others.

We can now sum up the rules of using RESTORE. In its simplest form, its syntax is this:

```
A>restore source target:filenames
```

Here the source is the disk containing BACKUP.001 and CONTROL.001. The target, whose directory path can be omitted, is the original directory from which files were backed up. File names must be the original, not BACKUP.001; wild cards can be used.

RESTORE's Switches

RESTORE has these switches:

/A:date	After; restore files backed up on or after date
/B:date	Before; restore files changed on or before date
/E:time	Earlier; files changed at or before specific time
/L:time	Later; files changed at or after specific time
/M	Modified; restore modified or deleted files
/N	None; restore files not in the target
/P	Prompt before replacing a changed file
/S	Subdirectory files are restored

The switches for date and time are the same as with the commands covered previously in this chapter. Just enter the numbers in the format as shown when you enter the DATE or TIME command. For the date, you must enter all six digits; for the time, you can omit the fractions and use the format shown with DIR.

The /M and /N switches differ slightly. If /N is used, only the files that do not exist on the target disk are restored. For example, if you erase FILE2 on the target disk and then enter this command:

```
A:\BACK>restore b: file? /n
```

only FILE2 will be restored. Since the other files are already on the target disk, they are not restored.

The /M switch will do the same thing as /N, plus modified files. For example, if you add a line to FILE2 on the target disk and then enter this command:

```
A:\BACK>restore b: file? /m
```

the version in the backup file will replace the modified version in the target drive. This will be done without any warning.

To prevent a modified file from being replaced by an old copy without any warning, you should use the /P switch, such as:

```
A:\BACK>restore b: file? /p
```

This will restore all the matching files. If there is a matching file in the target drive that has been modified, you will get this message:

```
WARNING! File FILE2
was changed after it was backed up
Replace the file (Y/N)?_
```

You can then press Y or N. Restoration continues if there are other files. You will get this same prompt several times if one file has been backed up several times using the `/A` switch.

Practice 7-4

☐ 1. Delete `TEST1` in drive A (created in the previous session). Restore it from the backup file.

☐ 2. Restore the remaining files from the backup with the `/P` and `/N` switches.

Drill

F 26. You can restore files to a directory different from the original source directory used by `BACKUP`. True or false?

D 27. The command `A:\BACK>restore b: backup.001` will:
a. restore all the original files
b. restore files not in the target
c. restore only modified files
d. generate an error message

T 28. The command `A>restore b: test? /n` will restore only the files that are in drive B but not in drive A. True or false?

Printing in the Background

✎ **Use `PRINT` to print files in the background while you continue to use your computer.**

So far we have discussed a variety of ways to print files, including Shift-Print Scrn to print a screen display, Ctrl-P to echo every move to the printer, and `COPY` to copy a file from a disk or the screen to the printer. These acts let the printer monopolize the entire system. While the printing goes on, you cannot use the computer.

If you want to use the printer and the computer at the same time, you need to use the `PRINT.COM` program. It provides a simple and crude multitasking mechanism. This program enables two devices (console and printer) to share the system. The CPU juggles between the two tasks and sometimes slows down one in favor of the other.

PRINT's Switches

PRINT has nine switches. Six can be specified only when you install the program; they set the parameters of the command. These are listed below:

/B:size	Buffer size in bytes, default 512
/D:dev	Device, such as COM1, default PRN
/M:max	Maximum number of clock ticks, 1–255, default 2
/Q:num	Queue; maximum number of files put in print queue
/S:slice	Time slices, 1–255, default 8
/U:wait	Wait clock ticks, 1–255, default 1

Three switches can be specified at any time after the program is installed; they are used to manage the print queue. These are shown below:

/C	Cancel printing file(s)
/P	Print file(s), default
/T	Terminate all printing

PRINT is partially memory-resident. After running it the first time, the resident portion stays in the memory until power off. If you want to enter the command again, you need to provide a path to where it is stored, just as for any external command.

To load this program with all the default settings, enter this:

```
A>print
Name of list device [PRN]:
Resident part of PRINT installed
PRINT queue is empty
```

When you enter PRINT the first time, you are asked what device you intend to use. If your printer is connected to a parallel port, the default of PRN is what you need. You can just press Enter to accept it. If you have something else, such as a serial port, enter a proper device name, such as:

```
A>print /d:com1
```

After you press Enter, a two-line message appears, telling you that the program is successfully installed and that there is no file in the print queue.

A **print queue** is a series of the names of the files you want to print. It is comparable to a series of numbers taken by people at a service counter waiting for their number to be called. You can line up files in the queue to be printed one after another. By default, the print queue size is set to 10 files. You can change it to 1 or up to 32 by using the /Q switch.

The /B switch can be used to increase or decrease the print buffer, which is set to 512 bytes by default. Increasing the print buffer speeds up printing but decreases available memory for DOS, just as increasing the value in the /Q switch does.

The /M, /S, and /U switches are used to determine how the computer and the printer share the time. When one gets more time, the other slows down. If you want the printer to work faster at the expense of the computer, increase the values of /M and /U or decrease the value of /S. Most of the time you have no need to tinker with them.

After PRINT is installed, you can use /P, /C, or /T to manage the print queue. The /P switch is the default and is often redundant. The /C switch is used to cancel individual files and the /T switch to terminate (empty) the entire print queue.

Steps to Print Files

To demonstrate how to use the PRINT command, we are going to print the files created earlier in this chapter. To conserve paper, we will not actually print all of them. Keep in mind that when PRINT prints a file, it consumes a whole page regardless of how short the file may be. We will selectively turn the printer on and off to demonstrate printing without wasting a great deal of paper.

If you have installed PRINT.COM as explained before, you are ready to go:

1. Turn off the printer if it is on.
2. Send files to the print queue:

```
A>print \back\file?

  A:\BACK\FILE1 is currently being printed
  A:\BACK\FILE2 is in queue
  A:\BACK\FILE3 is in queue
  A:\BACK\FILE4 is in queue
```

3. Clear the print queue:

```
A>print /t
Errors on list device indicate that it
may be off-line. Please check it.
PRINT queue is empty
```

4. Show the state of the print queue:

```
A>print
Errors on list device indicate that it
may be off-line. Please check it.
PRINT queue is empty
```

5. Send files to the print queue again:

```
A>print \back\file?
```

The same message shown after step 2 appears, plus a two-line error message as shown in steps 3 and 4.

6. Cancel three files and keep one:

```
A>print file1 /c file2 file4 file3 /p
Errors on list device indicate that it
may be off-line. Please check it.

  A:\BACK\FILE3 is currently being printed
  A:\BACK\FILE3 is in queue
```

In this arrangement, the /C switch affects one file before it and two files after it, and the /P switch controls a file placed before it. This command cancels FILE1, FILE2, and FILE4. The only file left in the print queue is FILE3.

7. Turn on the printer.

FILE3 is printed—sent from the print buffer to the printer. The paper rolls up one page. Turn off the printer if you want to prevent the paper from rolling up an entire page.

The /C and /P switches follow the same rule. Each controls the file placed immediately before it and all the files after it, unless negated by the other switch. Consider this example:

```
A>print /c file1 file2 file3 file4 /p
```

The first three files are controlled by the /C switch placed before them. The control on the fourth file, however, is negated by the /P switch placed after it. This command will result in clearing the first three files while retaining the last in the print queue.

PRINT operates in ASCII mode and reads a file only before the first ^Z. If a file contains such a character in the middle, such as our sample backup file discussed earlier, the portion after it is not printed.

If a backup file carrying a name like BACKUP.001 contains ASCII text, it can be printed like any text file, unless it contains a ^Z character in the middle. If a ^Z character is known to exist in it, you can use this to bypass it:

```
A>copy /b backup.001 prn
```

This will ignore ^Z and print everything that is printable.

Tips & Tricks

PRINT is a TSR

PRINT.COM, like APPEND.EXE, explained in Chapter 5, is another terminate-and-stay-resident (TSR) program. Once loaded, a portion of it stays in memory until power is shut off. Like APPEND, it may collide with other programs competing for the same memory area. It may clash with SHELL.EXE, which comes with the WordPerfect Library, and other similar shell programs.

If you load PRINT first and then run SHELL and other programs, you have no problem. If you run SHELL first and then load PRINT, your computer will freeze, requiring rebooting. Sometimes the fatal error message of Divide overflow also appears.

Practice 7-5

- ☐ 1. Turn off the printer. Send all the `TEST` files to the print queue.
- ☐ 2. Terminate the print queue.
- ☐ 3. Send `TEST1` and `TEST2` to the print queue again.
- ☐ 4. Cancel `TEST1` and `TEST2` but enter `TEST3` into the print queue. Turn on the printer to print `TEST3`.

(We have finished using all the files and directories created so far in this chapter. You can now erase or remove all of them.)

Drill

___ 29. The command `A>print test?` will send all the matching files to the print queue even if the printer is turned off. True or false?

___ 30. The command `A>print test1 /c test2 /p` will:
a. cancel the first file, print the second
b. cancel the second file, print the first
c. cancel both files
d. print both files

Summary and Review

Check each item as you review and understand the following highlights of the chapter:

- ☐ 1. Use `SYS.COM` to transfer the hidden system files of a new DOS version to your hard disk. Then use `COPY` to copy `COMMAND.COM`. When you reboot, the new version will be used.
- ☐ 2. Use `REPLACE` to replace the old DOS files on the hard disk. If you have version 4, use the `/U` switch to update the old files with the newer ones.
- ☐ 3. Use `ATTRIB` to set or remove A (archive) or `R` (read-only) attributes. The `A` attribute is automatically added when a file is created or modified. The `R` attribute can prevent a file's deletion, modification, or replacement.
- ☐ 4. Use `XCOPY` with `/M` to remove the A attribute of a source file after copying it. This can prevent another copying of the files that are not modified. You can, however, still use `XCOPY` with no switch to copy a file without the `A` attribute.

☐ 5. XCOPY cannot overwrite a file that is set to read-only. REPLACE, with /R, can overwrite such a file.

☐ 6. A target file of XCOPY and REPLACE has the A attribute automatically added, but the R attribute of a source file is not copied to the target.

☐ 7. Use BACKUP to back up important files on a regular basis. Backup files are numbered sequentially starting with BACKUP.001. They are also accompanied by sequentially numbered control files starting with CONTROL.001.

☐ 8. Using BACKUP without the /A switch causes the creation of a new backup file and erasure of all the files in the root directory of a target diskette. You need to have blank disks available to handle a large number of files.

☐ 9. Use the /M switch with BACKUP to back up only modified files. This will create a new backup file to contain only modified files. Modified files are those that have not been backed up. BACKUP removes a source file's A attribute after it is backed up. A new or modified file will have the A attribute, which the /M switch can recognize as needing backup.

☐ 10. Use the /A switch with BACKUP to add specific files to an existing backup file. The backup file must be available so that DOS can add files to it. Use both /A and /M to add modified files to an existing backup file.

☐ 11. If the original files are lost, you can use RESTORE to restore them from a backup file. The source is the backup file and the target the original directory, which was used as the source by BACKUP. You need to specify the original file names that you wish to restore.

☐ 12. You can restore files dated before or after a certain date or time. You can also restore files that are modified and/or do not exist in the target. If you use the /M switch, the target's modified file, which may be a newer version, will be replaced with its counterpart from the backup file. If you add /P, DOS will show you a matching file and ask for Y/N before replacing it.

☐ 13. Use PRINT to print files in the background. You can use a number of switches to determine how the printer and the computer are to share time. These switches must be specified when you run PRINT the first time. Most of the time you can just accept the default setting and do not need to tinker with these technical specifications.

☐ 14. Use /C to cancel specific files, /P to print specific files, and /T to terminate all printing. You can use these switches of PRINT at will, with the printer on or off.

Review Questions

Write an answer for each question below:

1. What is the purpose of `SYS.COM`?
2. What does this command do?

```
A>replace file? b:
```

3. What does this command do?

```
A>replace b:file? /a
```

4. What will happen if you insert a DOS disk in drive A and issue this command?

```
C:\>replace a:*.* /s
```

5. What does this command do?

```
A>attrib +r *.*
```

6. If both A and B contain `FILE1`, what will happen after the second command below?

```
A>attrib +r file1
A>replace b:file1 /r
```

7. If both A and B contain `FILE1`, what will happen after the second command below?

```
A>attrib +r file1
A>xcopy b:file1
```

8. What does `XCOPY`'s `/M` switch do?
9. What is the purpose of `BACKUP`? What files are created by it?
10. What does `RESTORE` do?
11. What does this command do?

```
A>backup c:\wp a:
```

12. What does this command do?

```
C:\WP>backup file? a: /m/a
```

13. How does `BACKUP` recognize a modified file?
14. What does this command do?

```
A>restore b: file? /n
```

15 What does this command do?

```
A>print /c file1 file2 file3 /p
```

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
System files	Must be the first entries and occupy contiguous sectors	Can be fragmented or placed in other sectors
SYS	Cannot specify a source drive for system files	Can specify a source drive for system files
REPLACE /U	Switch not available	New switch
BACKUP /F	Switch needed if target is to be formatted first	Switch not needed; target automatically formatted if needed

Redirecting Files

Chapter 8

TOPICS:

Redirecting Output
Redirecting Output with ECHO
Redirecting Input
Pipes and Filters
Using MORE
Using SORT
Using FIND
Combining Filters
Inserting Date and Time

COMMANDS COVERED:

ECHO, MORE, SORT, FIND, FOR..IN..DO

NEW TERMS AND SYMBOLS:

standard input device, standard output device, redirection, pipe, >, >>, <, |, filter

REQUIRED MATERIALS:

1. Two practice disks
2. Access to MORE.COM, FIND.EXE, SORT.EXE, CHKDSK.COM, and ATTRIB.EXE

Quick Start

✎ **For piping and redirection symbols:**
See Table 8.1.

✎ **To redirect directory output to a file:**

```
A>dir > filename
```

✎ **To print directory entries:**

```
A>dir > prn
```

✎ **To redirect file display to the printer:**

```
A>type filename > prn
```

✎ **To respond to some DOS commands, redirect input from a file:**

```
A>del *.* < filename
```

✎ **To print a text line entered from the keyboard:**

```
A>echo This line is printed > prn
```

✎ **To display directory entries one screen at a time:**

```
A>dir ¦ more
```

✎ **To display a text file one screen at a time:**

```
A>more < filename
```

✎ **To print a typed text line as soon as Enter is pressed:**

```
A>more > prn
```

✎ **To sort the text lines of a file by first word:**

```
A>sort < filename
```

✎ **To sort directory entries by extension and one screen at a time:**

```
A>dir ¦ sort /+10 ¦ more
```

✎ **To find "Smith" lines from a file:**

```
A>find "Smith" filename
```

✎ **To find "Smith" lines but not show 1-inch line and file name:**

```
A>find "Smith" filename ¦ find "Smith"
```

✎ **To find "SMITH" files, sorted and displayed one screen at a time:**

```
A>dir ¦ find "SMITH" ¦ sort ¦ more
```

✎ **To find files and directories on a disk matching "SMITH":**

```
A>chkdsk /v ¦ find "SMITH" ¦ sort ¦ more
```

✎ **To insert date and time in a file:**

```
A>echo ¦ more ¦ date ¦ find "C" >> filename
A>echo ¦ more ¦ time ¦ find "C" >> filename
```

Normally DOS works under certain presumptions. Input (raw data) is expected to come from the **standard input device**, which is the keyboard. Output (processed data) goes to the **standard output device**, which is the monitor screen. Error (what went wrong) and informational (what has been done) messages also go to the standard output device.

Any time you want to deviate from this pattern, you must tell DOS to **redirect** input and/or output to another device. Input can be redirected from a file saved on disk or from a device such as a mouse, modem, or scanner. Output can be redirected to a number of devices, such as disk drive, printer, modem, and so on.

Besides the ability to redirect input and output, DOS provides the mechanism to **pipe** (connect) several commands together. Furthermore, three filter commands are provided, giving you more power to maneuver your files.

Redirection uses the first three symbols shown in Table 8.1. The fourth is used for piping.

Table 8.1 Redirection and Piping Symbols

Symbol	Purpose
>	Redirect to a new output device
<	Redirect from a new input device
>>	Append to an output device
¦	Pipe (chain) commands

This chapter first explains redirection and then connects (pipes) various versatile tools together with the filter commands of MORE, SORT, and FIND.

Redirecting Output

✎ **Use > or >> with various DOS commands to redirect or append outputs to devices or files.**

This section demonstrates how to redirect output and compares output redirection with the COPY command.

Redirection Steps

Turn on the printer, insert a practice disk in drive A and another disk in drive B, and follow these steps to redirect output to new devices:

1. Output current directory entries to a file on disk:

   ```
   A>dir > fildir
   ```

   Instead of the usual display on the screen, nothing shows up. Drive A is activated and a file named `FILDIR` is created. If this file already exists, its original contents will be wiped out.

2. Display on the screen the file just created:

   ```
   A>type fildir
   ```

   The screen shows what you will normally see when `DIR` is entered and no redirection is involved.

3. Print the file (redirect it to the printer):

   ```
   A>type fildir > prn
   ```

   The file is now printed. Normally, when you use `TYPE`, a file's contents appear on the screen. The above command redirects the output to the device named `PRN`, which could also be `LPT1` (or `COM1` if your printer is connected to the first serial port).

4. Append drive B's directory entries to `FILDIR`:

   ```
   A>dir b:\ /w >> fildir
   ```

   This adds the wide-format directory entries of the root directory in drive B to `FILDIR`. Do not use a single > symbol here. If you do, the new data will replace the old.

5. Show the expanded file on the screen:

   ```
   A>type fildir
   ```

   If you did everything correctly, the screen should show a display consisting of a long part at the beginning and a wide part at the end.

`COPY` and Output Redirection

The command in step 3 above could have been this instead:

```
A>copy fildir prn
```

This will also print the file. The only difference is the display of this message:

```
1 File(s) copied
```

If you use COPY, however, do not use the > symbol between the source file and the target device, such as:

```
A>copy fildir > prn
```

This command will neither print the file nor make a duplicate of it. DOS will consider your move to be copying a file onto itself, which is illegal.

If you intend to copy a file and redirect the message `1 File(s) copied` to the printer, here is a legal way:

```
A>copy fildir fildir2 > prn
```

If you intend to redirect a file to the NUL device, in effect not doing anything or showing any result on the screen, both commands below will work:

```
A>type fildir > nul
A>copy fildir nul > nul
```

The second command will copy the file to NUL and redirect any message to NUL as well. If the second NUL (together with the > symbol) is missing, the `1 File(s) copied` message will appear on the screen. If the first is missing, the copying will fail due to the absence of a target.

You already know that instead of TYPE you can use the following command to display a file's contents:

```
A>copy fildir con
```

This will display the file, plus the message `1 File(s) copied`, which is what distinguishes TYPE and COPY CON. You can redirect the message to the NUL device from the screen:

```
A>copy fildir con > nul
```

This will make COPY CON behave exactly like TYPE. You can also redirect the message to the printer:

```
A>copy fildir con > prn
```

Tips & Tricks

Error Messages Cannot Be Redirected

DOS produces two kinds of messages. Informational messages tell you what has been or will be done. Error messages tell you what has gone wrong. The former can be redirected to a device. The latter, however, always go to the screen and cannot be redirected. Consider the following two examples:

```
A>copy filename
        File cannot be copied onto itself
                0 File(s) copied

A>copy filename > prn
File cannot be copied onto itself
```

In the first case, both lines are displayed on the screen. In the second, the `0 File(s) copied` message goes to the printer, but the error message remains on the screen.

Redirecting Output with ECHO

✎ **Use ECHO to redirect data to a file or the printer.**

The ECHO command is normally used to echo data to the monitor screen, the standard output device. You can use it to redirect screen data to various other places. Turn on your printer and enter the following commands to see what each does:

1. `A>echo This is a test.`
2. `A>echo This is a test. > con`
3. `A>echo This is a test. > nul`
4. `A>echo This is a test. > prn`
5. `A>echo This is a test. > testfile`
6. `A>echo This is a test. >> testfile`
7. `A>echo >> testfile`
8. `A>echo < testfile`
9. `A>echo This line is ">" displayed on the screen.`
10. `A>echo This is '>' a file.`
11. `A>echo con > prn`

The first and second lines behave the same way, echoing the text line to the standard output device, namely the screen.

Lines 3 and 4, respectively, redirect output to NUL (discarding it) and the printer (printing it).

Line 5 creates a file and saves to it the text between ECHO and >; if the file exists, its original contents are replaced. Line 6 appends the text to an existing file, without replacing its contents. Line 7 appends the ECHO is on message to the named file. Line 8 displays on the screen the ECHO is on message, thus ignoring the input redirection.

To display a redirection or piping symbol, you need to put it in double quotes. In line 9, everything after ECHO is echoed to the screen as originally entered. In line 10, a file with a single quote (the first word located right after >) as its name will be saved. Its contents consist of this line:[1]

```
This is ' a file.
```

Line 11 sends con to the printer. You might interpret this line as letting you enter input on the screen and then output it to the printer. It does not work that way; ECHO does not accept input from a file (such as line 8) or a device such as CON.

ECHO is commonly used in batch files and with ANSI.SYS. These are covered in Chapters 11 and 12.

Redirecting Input

✎ **Use the < symbol to redirect an input from a file, which may provide a necessary response to a DOS command.**

There are occasions when a DOS command needs an input before making its next move. Most of the time, you are asked to enter Y or N from the keyboard. Instead of entering these manually, you can redirect them from an existing disk file that contains a predetermined input.

Input for DEL

An example of the need to enter a Y/N input is when you use this command:

```
A>del *.*
```

As you already learned, this command will lead to this response:

```
A>del *.*
All files in directory will be deleted!
Are you sure (Y/N)?_
```

If you wish to go ahead, you are supposed to type Y and then press Enter to start deleting.

If you know what you are doing, you can simplify your work with this command:

```
A>del *.* < yes
```

[1] The odd thing about this is that the contents both before and after the redirection symbol are sent to the file.

Here YES is a file name intended to supply the Y response when asked by DOS. You now need to create this file named YES, as shown below:

```
A>copy con yes
y
^Z
```

This file contains two keystrokes, Y and a carriage return. Do not forget to press Enter after Y. It is needed to get the DEL command going.

You can put the whole thing together in a batch file like this:

```
A>copy con delall.bat
del *.* < yes
^Z
```

When you run this batch file, DOS automatically goes through these steps:

```
A>delall
del *.* < yes
All files in directory will be deleted!
Are you sure (Y/N)?y
```

The Y, followed by Enter, is supplied by the YES file. So DOS starts deletion without further delay.

Warning: This batch file is a mass file killer. Do not use it unless you know what you are doing. When you run it, by simply entering DELALL, all the files in the current directory will be immediately wiped out. If the batch file is also stored there, it will also disappear; as a result, the Batch file missing message will be displayed.

If you want to experiment with this batch file, you can create a separate directory, make it the current directory, and put some dispensable files there. Provide a path to the root directory where you store DELALL.BAT. Run the batch file and see what happens.

Instead of creating two separate files as illustrated above, you could integrate the whole thing into one file like this:

```
A>copy con delall.bat
echo y > yes
del *.* < yes
^Z
```

Each time this modified batch file is run, the first line will create a new file named YES which will contain a single character of Y, plus a carriage return; if this file already exists, it will be replaced. The second line will redirect the required Y from the newly created file to delete everything, including the newly created YES file.

You may ask, why bother redirecting input? There is a simpler way, you say. Here is what you might contemplate:

```
A>copy con delall.bat
del *.* y
^Z
```

Another possibility you might have in mind is this:

```
A>copy con delall.bat
del *.*
y
^Z
```

Neither one will work. The first will generate an error message—too many parameters. The second will wait for your response and then run a program or batch file named Y, which will most likely generate an error message—bad command or file name.

Input for FORMAT

Although supplying Y blindly is a dangerous practice and defeats the purpose of letting you exercise caution before a destructive move, supplying N could be time saving. Consider the following batch file:

```
A>copy con fmat.bat
@echo off
c:\dos\format b: < go
^Z
```

This batch file is intended to load FORMAT.COM from the C:\DOS directory and then format the disk in drive B. The required responses are input from the GO file, which is created with the following steps:

```
A>copy con go


n
^Z
```

This file has two empty lines (pressing Enter twice) and one letter N. The first Enter is to respond to this:

```
Insert new diskette for drive B:
and press ENTER when ready...
```

The second Enter is to enter no volume label. After formatting is complete, you are prompted to enter a volume label. This requires a response from you. By pressing Enter (or redirecting the required input from a file), no volume label is created, and the formatting process moves on. You can, if you wish, enter a single label here, such as ABCCORP; in that case, all the disks formatted with this batch file will carry this label.

The N in the GO file is to tell DOS to format no more. This is the last response required to complete a formatting process.

Practice 8-1

- ☐ 1. Use COPY CON to create a file named XYZ; enter a few lines of text.
- ☐ 2. Use ECHO to add a line to XYZ.
- ☐ 3. Use TYPE to send XYZ to NUL and then to the printer.
- ☐ 4. Repeat the above steps with COPY.
- ☐ 5. Use COPY CON to create a file named CR that contains a single carriage return (pressing Enter once). Create a batch file named T.BAT that will redirect CR to the TIME command and thus allow you to enter T alone to show the current time.

Drill

_____ 1. The command `A>dir > file1` will:
   a. create a file named `FILE1`
   b. destroy `FILE1`'s contents if it exists
   c. do neither a nor b
   d. do both a and b

_____ 2. The command `A>dir >> file1` will:
   a. append data to `FILE1`, if existent
   b. replace the contents of `FILE1` if existent
   c. create `FILE1` if it does not already exist
   d. do both a and c
   e. do none of the above

_____ 3. The command `A>type file1 > prn` will:
   a. display and print `FILE1`
   b. print `FILE1`
   c. display `FILE1`
   d. do nothing

_____ 4. The command `A>copy file1 > prn` will:
   a. print `FILE1`
   b. copy `FILE1` to a file named `PRN`
   c. do both a and b
   d. generate an error message

_____ 5. If your printer is turned on, the command `A>echo test line > prn` will:
   a. print two words
   b. print three words
   c. do nothing
   d. generate an error message

_____ 6. The command `A>copy file1 nul > nul` will:
   a. copy `FILE1` to a file named `NUL`
   b. display a message on the screen
   c. copy to device `NUL` and show no message
   d. generate an error message

_____ 7. If `YES` is not a file, the command `A>del *.* < yes` will:
   a. delete all files immediately
   b. prompt you to enter `Y/N`
   c. do nothing
   d. generate a `File not found` message

_____ 8. To enable the command `A>format b: < go` to complete its job without any human intervention, the `GO` file must contain:
   a. a single N
   b. a blank line, followed by N
   c. two blank lines, N, followed by Enter
   d. a blank line, two Ns

Pipes and Filters

✎ **Use a pipe to connect two DOS commands.**

✎ **Use a filter command to sort, find, or display files.**

So far you have run each DOS command by itself. Here you will learn that several DOS commands can be chained together and input and output can be piped from one to another.

This section explains the general principles of piping and filtering. Details of related commands will follow.

Piping

A **pipe** (¦) is a connector of two commands. Like a water pipe (or hose) that connects a faucet to a garden, a DOS pipe connects two commands or programs placed before and after it. When you turn on the faucet, the water that comes out of the faucet goes through the pipe to the garden. The output from the command before a DOS pipe is fed to the next command as input. Consider the following arrangement:

```
A>dir ¦ find "EXE" ¦ sort
```

The output of `DIR` is fed to `FIND`. When matching files are found, they are fed to `SORT`, which displays the found files in alphabetical order.

Pipes can be used to run one program after another, such as:

```
A>prog1 ¦ prog2 ¦ prog3
```

After you exit `PROG1`, `PROG2` is executed, and so on.

If you pipe two DOS commands together, in which the first does not produce an output that can be used as an input for the second, the first output is not displayed and only the second is, such as:

```
A>dir ¦ path
A>path ¦ dir
```

In the first example, `DIR`'s output is not shown and only the output by `PATH` is displayed; in the second, the reverse is true.

In a piping operation, DOS saves the output of a command to the root directory of the default drive, and then uses the file as input to the next command. The file is then erased. On some occasions, you may find such remnants with odd names, as shown below:

```
132A0A53                0  06-15-90    7:42p
132A0A59                0  06-15-90    7:42p
```

The file names are hex numbers based on the internal clock. If you find these empty files, erase them.

Filtering

A **filter** is a device that materials go through and come out different from their original state. DOS provides three filter commands:

`MORE`	Display text 24 lines at a time
`SORT`	Produce sorted output
`FIND`	Display found lines

These filters process raw materials and produce the outputs explained above. They can be used alone or in combination with others. Their output can go to the screen or be redirected to other devices.

Using `MORE`

- **Redirect an input file to `MORE` to display one screen at a time.**
- **Use `MORE` and redirect output to `PRN` to print each typed line.**

The `MORE` command filters from a source and redirects output to the screen one screen (page) at a time. Twenty-four lines are displayed at one time. The bottom of the screen displays this message:

```
- More -
```

It tells you that there is more. Pressing a key will display another page until the end. Pressing Ctrl-C or Ctrl-Break will terminate the display.

Using `MORE` Alone

If the command is entered alone, you may be puzzled about its behavior. The cursor moves down two lines and nothing else happens:

```
A>more


_
```

You might think the computer is locked up. It is not. If you press Enter alone, the cursor keeps moving down two lines each time and nothing else happens. If you enter a line, it is immediately echoed (duplicated) to the screen.

This behavior may seem puzzling to you, but it is actually logical. Without given any direction, DOS assumes that you are inputting from the keyboard (standard input) and outputting to the screen (standard output). So every time you type something and press Enter, it is duplicated on the screen. To get out of the MORE screen, press Ctrl-C or Ctrl-Z and Enter.

Using MORE to Display

MORE is most often used to output text lines to the screen one screen at a time. The text lines could be directory entries or input from a disk file. The syntax for each situation appears like this:

```
A>dir ¦ more
A>more < filename
```

In the first case, the output of DIR is filtered by MORE. If the display goes beyond a screenful, MORE will pause and display a message. The result is comparable to using the /P switch with DIR. In the second case, the disk file is input to the MORE filter and regulated the same way.

The symbols may look confusing. Can other symbols be used? Yes. However, each symbol produces a different result. The list below shows all possible variations:

```
A>more ¦ dir       Execute MORE and then DIR
A>more > dir       Create a file named DIR
A>more >> dir      Append to a file named DIR
A>more < dir       Input a file named DIR
A>more << dir      Same as above
A>dir > more       Create a file named MORE
A>dir >> more      Append to a file named MORE
A>dir ¦ more       Filter directory through MORE
A>dir < more       MORE file ignored or File not found
A>dir << more      Same as above
```

In the first example, the MORE command is executed without any parameter. After you enter ^Z to end, the DIR command is then executed. If you enter ^C, DIR is not executed.

In the second and third examples, MORE is executed to allow you to enter text lines. When you enter ^C or ^Z, the text lines are sent to a file named DIR (this has nothing to do with the DIR command).

The last two examples do not make sense because the DIR command does not accept an input from a disk file. If the file named MORE (not MORE.COM) exists, DIR is executed but the file input is ignored. If the file does not exist, the File not found message is shown and the DIR command is not executed.

MORE, TYPE, and COPY CON

In displaying file contents on the screen, the MORE command behaves very much like TYPE, with the exception of showing one page at a time. They both treat a file as ASCII text and stop reading it as soon as the first ^Z (end-of-file marker) is encountered. Follow these steps to illustrate this point:

1. Use MORE to display MORE.COM (make sure MORE.COM is in drive A or accessible through an APPEND list):

```
A>more < more.com
```

2. Use TYPE to display MORE.COM:

```
A>type more.com
```

3. Display the entire MORE.COM in the binary format:

```
A>copy /b more.com con
```

The first two steps show the same result, a short display of the program code. The third shows a much longer program. The role of ^Z was explained in Chapters 4 and 6.

How about showing an entire binary file one screen at a time, like this:

```
A>copy /b more.com con ¦ more
```

Unfortunately, it will not work. Since COPY CON produces no output that MORE can use, the MORE command is ignored.

MORE will also work with TYPE to show a text file one screen at a time:

```
A>type filename ¦ more
```

This will work the same as:

```
A>more < filename
```

Using a Batch File to Display Files

If you want an easy way to display file contents, the short batch file below will work well:

```
A>copy con seemore.bat
@echo off
more  %1
^Z
```

When you want to show a file, just enter a command like this:

```
A>seemore filename
```

This requires you to enter the name of a file to be displayed. This name is then assigned to the %1 replaceable parameter. If no file name is entered, this message appears:

```
File not found
```

If this happens, MORE is not loaded—unlike the situation when the MORE command is entered alone.

Using MORE to Print

MORE can be used to print text lines entered from the keyboard. This will add another printing technique to what you learned in Chapter 2. Now turn on the printer and follow these steps to see what happens.

1. Redirect output to the printer:

   ```
   A>more > prn
   ```

2. Enter a line:

   ```
   This line goes to the printer.
   ```

   What happens? Instead of outputting the line to the screen, DOS redirects it to the printer.

3. Enter a line after pressing Tab once:

   ```
   This line is indented.
   ```

   The line is printed at the first tab. The print head moves down one line but remains in the same column instead of returning to the left margin.

4. Press Enter alone. The cursor moves down one line and the printer roller moves up one line.

5. Press F7 (to display ^@) and Enter to send a null character. The above phenomenon is repeated.

6. Send ASCII 12 by pressing Ctrl-L, or hold down Alt and type 12 on the numeric keypad. This is the form-feed character; it will cause the paper to advance by one page.

7. Press Ctrl-P, enter a line, and press Ctrl-P again. Each letter is sent to the printer as it is typed. When you press Enter, the whole line is printed again, with each letter appearing twice.

8. Press Ctrl-C (or F6 and Enter to enter ^Z) to terminate.

The above steps show that with MORE every line is faithfully redirected to the printer. At step 8, either method produces the same result. There is nothing left to print, and nothing is printed.

In contrast, repeat the above steps after entering this command:

```
A>copy con prn
```

Nothing is printed until you press Ctrl-P. Entering ^Z at step 8 sends everything to the printer. Entering ^C (pressing Ctrl-C) prints nothing.

Tips & Tricks

Printing from the Screen

If you need to print a short document you type on the screen, there are a number of ways to do that.

For printing each line after Enter is pressed, use this command:

```
A>more > prn
```

For printing each letter as soon as you press it, follow these steps:*

1. Enter this command to begin:

```
A>copy con prn
```

2. Press Ctrl-P and start typing.
3. Press Ctrl-P and then Ctrl-C when done.

To print after the whole document is finished, follow these steps:

1. Enter this command to begin:

```
A>copy con prn
```

2. When done, press F6 and Enter to print.

Practice 8-2

- ☐ 1. Use `MORE` to type and print three lines of text.
- ☐ 2. Use `MORE` to redirect `XYZ` (created in the previous session) to `NUL`.
- ☐ 3. Use `MORE` to redirect the above file to the printer.

Drill

_____ 9. The command `A>path¦dir` will:
   a. display a path list
   b. display directory entries
   c. display both
   d. generate an error message

*Again, there will be some varience between daisy-wheel and dot-matrix printers. See p. 55.

_____ 10. The command `A>dir ¦ more` will display directory entries one screen at a time. True or false?

_____ 11. The command `A>dir > more` will do the same as in Question 10 above. True or false?

_____ 12. If a file named MORE exists in the current directory, the command `A>dir < more` will:
a. display the current directory all at once
b. display the file named MORE a screen at a time
c. create a file named MORE
d. display `File not found` message

_____ 13. The MORE filter can display the contents of an entire binary file. True or false?

_____ 14. The command `A>more > prn` will print each text line as it is entered on the screen. True or false?

_____ 15. If MORE is entered without a parameter, an error message will be displayed. True or false?

Using SORT

- **Use SORT to sort text lines of an ASCII text file.**
- **Use the /R switch to sort in descending order; use the /+c switch to sort by a specific column.**
- **Use DIR with SORT and a column number to display sorted directory entries.**

The SORT filter can sort text lines that go to the screen. These text lines can include those you enter from the keyboard, those input from a disk file, or directory entries.

Creating and Sorting a Database

Before demonstrating how the SORT command works, let us first create a database of quotations and then manipulate it in a variety of ways. Make sure SORT.EXE is in drive A or accessible somewhere before proceeding further.

1. Create a database by entering the following:

```
A>copy con quotes
Confucius: He who does not economize will have to agonize.
Franklin: He who has a trade has an estate.
Lao Tse: He who knows does not speak.
Reagan: We start bombing Russia in five minutes.
Cervantes: Be slow of tongue and quick of eye.
^Z
```

2. Sort the file:

```
A>sort < quotes
Cervantes: Be slow of tongue and quick of eye.
Confucius: He who does not economize will have to agonize.
Franklin: He who has a trade has an estate.
Lao Tse: He who knows does not speak.
Reagan: We start bombing Russia in five minutes.
```

To sort a file by first word, simply input it to the SORT filter. The file is now sorted in ascending order.

3. Save after sorting:

```
A>sort < quotes > quotes.srt
```

If you want to save the sorted file, simply redirect the output to a file name, as above. The screen does not display the sorted file. Instead, it is redirected to the named file. If you use the same file name for both input and output, the sorted version replaces the original.

4. Sort in descending order and append to an existing file:

```
A>sort /r < quotes >> quotes.srt
```

The target file now contains duplicates. The top portion is in ascending order and the bottom portion the reverse of that.

Sorting by a Column

In addition to /R (reverse), SORT has another switch, /+c. To use this switch, put a plus sign after the backslash and a column number, which is the position of a letter counting from the left margin.

To demonstrate how this works, let us build another database that can be meaningfully sorted according to a column number, and then sort by that column:

1. Create the file below; press Tab twice before entering each number:

```
A>copy con phones
Townsend, Skip          453-8909
Zagler, Zig             323-2121
Bumsteer, Dag           405-3216
Wane, Johnny            987-6543
Wayman, Jane            555-0101
^Z
```

2. Sort by the 24th column:

```
A>sort < phones /+24
Zagler, Zig             323-2121
Bumsteer, Dag           405-3216
Wane, Johnny            987-6543
Wayman, Jane            555-0101
Townsend, Skip          453-8909
```

The result is not correct. Each phone number appears to begin in the 24th column. It actually begins in the 17th.

3. Sort by the 17th column:

```
A>sort < phones /+17
Zagler, Zig              323-2121
Bumsteer, Dag            405-3216
Townsend, Skip           453-8909
Wayman, Jane             555-0101
Wane, Johnny             987-6543
```

This time the numbers appear in correct ascending order.

In counting a column number, keep in mind that a space is a column, and so is a tab. In our database, the longest name has 14 characters (columns). Since we press Tab twice after each name, that makes each number appear after 14 + 2 columns, namely in the 17th. (It is better to separate fields with spaces rather than tabs.)

Sorting Directory Entries

SORT can be used to display your directory entries in a way that will enable you to find files more quickly. This command will display all files in ascending order:

```
A>dir ¦ sort
```

If you have many files in a directory, the above command is most useful. Since all the file names are shown in alphabetical order, you can quickly locate what you are looking for.

You can also enlist the help of MORE, such as:

```
A>dir ¦ sort ¦ more
```

One screen of files will be shown at a time, in alphabetical order.

Directory items of each file are arranged in a logical manner. You can use this orderly arrangement to sort by a specific column. The following and Table 8.2 show where each item is located:

```
CHAP       001          3547 09-17-89   10:30p
CHAP       002         12317 09-27-89    2:34p
CHAP       003      11101117 01-09-90    8:01p
```

Table 8.2 Directory Entries and Column Positions

Directory Entry	Column
File name	1
Extension	10
File size	14
Month	24
Date	27
Year	30
Hour	34
Minute	37

To sort by extension, enter this command:

```
A>dir ¦ sort /+10 ¦ more
```

All the items shown, including the messages and file counts, are sorted. First come the files without extensions, the total number of which is tabulated and displayed. Then come subdirectories, if any. Finally appear other extensions in alphabetical order, with messages mixed together.

This command will sort by file size:

```
A>dir ¦ sort /+14 ¦ more
```

Sorting by different date/time items is not particularly meaningful. If you go by month (/+24), you will end up having this order:

```
CHAP       003   11101117 01-09-90    8:01p
CHAP       001       3547 09-17-89   10:30p
CHAP       002      12317 09-27-89    2:34p
```

Going by year (/+30) produces the least misleading order. The month and date will not be in order, but the year will be correct.

Numeric vs. Alphanumeric Sort

SORT can sort alphanumeric words accurately. The ASCII values are compared, but uppercase and lowercase are adjusted (placing a, rather than B, after A). To demonstrate how SORT works with entries from the keyboard, follow these steps:

1. Run the SORT program without entering any parameter.
2. Enter from the keyboard the items to be sorted.
3. Enter ^Z by pressing F6 and Enter.

The results of the above steps are shown below:

```
A>sort
Zip
zap
Abel
ace
able
^Z
Abel
able
ace
zap
Zip
```

SORT cannot do an accurate numeric sort. Numbers are treated as alphanumeric characters whose values are based on the ASCII table, except adjusting for uppercase and lowercase. The following is an example:

```
A>sort
111119
11118
9.999
0.2222
0.2221
^Z
```

```
0.2221
0.2222
111119
11118
9.999
```

Numbers of unequal lengths are not placed according to their numeric values. It is thus meaningless to sort such numbers. However, numbers of equal lengths, such as ZIP, license, telephone, or Social Security numbers can be sorted accurately and quickly.

SORT's Input and Output

SORT's output can be redirected to a device such as a printer. Here are two possibilities:

```
A>sort > prn
A>sort /r < filename > prn
```

The first line allows you to enter on the screen the items to be sorted. When ^Z is entered, the sorted lines are sent to the printer, not the screen.

The second example sorts the named file in reverse order. The output is also redirected to the printer rather than the screen.

Practice 8-3

☐ 1. Build a database of 12 records and name it ADDRESS. Each record should have four fields as shown below. The State field should contain four states, each of which should appear in three records.

```
Drake, Daffy        123 Kwack Ave., Fun City        CA    94000
```

☐ 2. Sort the records by last name in ascending order.

☐ 3. Sort by last name in descending order.

☐ 4. Sort by state in ascending order.

☐ 5. Sort by ZIP in ascending order and print the output.

Drill

_____ 16. The SORT command can accurately sort numerals of unequal lengths. True or false?

_____ 17. SORT strictly follows the ASCII table in arranging alphanumeric characters. True or false?

_____ 18. In determining which column to sort by, SORT treats a tab as a column. True or false?

_____ 19. If you have a database in which the first field contains first name followed by last name, you can use SORT to accurately sort by last name. True or false?

_____ 20. By default, SORT sorts by descending order. True or false?

_____ 21. SORT can be used to sort directory entries by any specific column. True or false?

_____ 22. The command `A>dir¦sort¦more` will:
a. display 24 lines, sorted
b. display the —MORE— message if there is more
c. generate an error message
d. do both a and b

Using FIND

✎ **Use FIND with a search string to find text lines in a file.**

✎ **Use /V to find lines not containing a search string, /C to find the number of matching lines, and /N to number and display matching lines.**

The FIND command allows you to search for text strings in a file or directory entries. Input data going to the screen is filtered by FIND and output is displayed or redirected.

Finding Text Strings

To find a text string from a disk file, enter a command like this:

```
A>find "string" filename
```

The string you want to search for has to be put inside double quotes. If the string is found, the whole line containing it is displayed on the screen.

Let us try to find a quote in the QUOTES file created earlier. Before proceeding further, make sure that FIND.EXE is either in drive A or accessible somewhere:

1. Enter a word in lowercase as a search string:

```
A>find "franklin" quotes

---------- QUOTES
```

DOS tells you that nothing matching is found. The file name is displayed, but nothing else is shown.

2. Use a different case:

```
A>find "Franklin" quotes

---------- QUOTES
Franklin: He who has a trade has an estate.
```

This time, one matching line is found. The difference this time is using uppercase for the first letter, which matches the entry in the database. So, keep in mind that a search operation is case sensitive.

3. Enter a phrase as a search string:

```
A>find "He who" quotes
```

This will catch three lines. A search string can be anything, as long as it is inside quotes. As shown here, you can use several words as a search string.

4. Enter two search strings:

```
A>find "He" quotes | find "Confucius" quotes

---------- QUOTES
Confucius: He who does not economize will have to agonize.
```

This tells DOS to search for lines containing "He" and then search in that group for lines containing "Confucius". It will catch only one line, together with the file name, as shown above.

5. Omit the file name in the second search string:

```
A>find "He who" quotes | find "Confucius"
Confucius: He who does not economize will have to agonize.
```

The same result appears, but the file name is not displayed.

Tips & Tricks

FIND without Line or File Name

When you use the FIND command to search for lines in a file, the screen always shows a one-inch line, followed by the file name in uppercase.

Some people go through a lot of trouble devising complicated ways to remove the distracting line and file name. If you find these items objectionable and do not want them displayed, the simplest trick is to repeat a search string without the file name, such as:

```
A>find "He who" quotes | find "He who"
```

This in effect suppresses display of the extraneous line. The extra line does not appear regardless of whether any matching line is found or not. This technique can save you a lot of paper if you redirect output to the printer, such as:

```
A>find "He who" quotes | find "He who" > prn
```

FIND's Switches

FIND has three switches, as shown below:

/V	Void; shows lines not containing search string
/C	Count number of lines found
/N	Number; a number precedes each found line

To find the lines not containing "He who", enter this command:

```
A>find /v "He who" quotes
```

This will show two lines.

When you want to know how many lines without actually displaying the lines, use the /C switch. If you want each line found displayed and numbered, use the /N switch, such as:

```
A>find /n "He who" quotes

---------- QUOTES
[1]Confucius:He who does not economize will have to agonize.
[2]Franklin: He who has a trade has an estate.
[3]Lao Tse: He who knows does not speak.
```

When /C is used, alone or with /V and/or /N, no line is displayed; only the total number of matching lines is shown. In effect, /C negates the other two switches.

Finding Directory Entries

FIND can be used to find directory entries, such as:

```
A>dir | find "EXE"
```

The output of DIR will be filtered by FIND in search of any line containing the search string of "EXE". This command will thus display all the files in the current directory with the EXE extension, plus possibly other matching files. Do not put a period in the search string because no period appears anywhere on the screen produced by DIR. If you do not want subdirectories displayed, this command will do:

```
A>dir | find /v "<DIR>"
```

The search string in the above situation could be shortened to "<".

FIND, like MORE and SORT, reads a file as ASCII text. It stops reading when the first ^Z is encountered. Consequently, it is not very useful in finding text strings in binary files.

Combining Filters

✎ **Combine `FIND`, `SORT`, `MORE`, `CHKDSK`, and `ATTRIB`, and put them in batch files to quickly search for text lines or file names.**

Most of the data output to the screen can be filtered and redirected, including file names and text lines. This section shows how to combine various filter commands to produce outputs in specific ways.

Before file names are output to the screen, you can use a number of ways to filter them, such as this example:

```
A>dir ¦ find "REPORT" ¦ sort /+30
```

This command will list all the file names carrying the letters of `REPORT`, such as `REPORT1.TXT` and `NWREPORT`. These files are also sorted by their 30th column, namely year.

Warning: When filtering file names, make sure you use uppercase in a search string because DOS outputs file names only in uppercase. If you were to use `"report"` as a search string, nothing would be shown because no file name is displayed in lowercase.

Since `CHKDSK` and `ATTRIB` also output file names to the screen, you can also filter the outputs. These have one advantage over `DIR`: They can handle sub-directories. Consider the following two examples:

```
A>chkdsk /v ¦ find "SMITH" ¦ more
A>attrib /s *.* ¦ find "SMITH" ¦ more
```

Both produce file names and their directory paths. The only difference is that the latter also displays file attributes. The `MORE` filter is necessary only if you expect lots of `SMITH` files.

Hard Disk Users: If you have complex directories and spend a lot of time searching for files, this batch file should be helpful:

```
C:\BAT>copy con seefile.bat
@echo off
path=c:\dos
chkdsk /v ¦ find "%1"
^Z
```

When you run this batch file, make sure you use uppercase in a search string, such as:

```
C:\BAT>seefile SMITH
```

The batch file will make DOS go to `C:\DOS` to look for `CHKDSK` and `FIND`. The last line in the batch file will be executed this way:

```
chkdsk /v ¦ find "SMITH"
```

The entire drive C will be searched and all the `SMITH` files (and directories) and their paths will be displayed.

If you need to look for text lines inside various files, the batch file below should be useful (notice the double %% inside a batch file):

```
A>copy con seeline.bat
@echo off
for %%x in (%1) do find "%2" %%x
^Z
```

You need to enter two parameters when you use this batch file, the first for file name(s) and the second for search string, like these two examples:

```
A>seeline file? Smith
A>seeline *.* Smith
```

The first searches for "Smith" in files matching the FILE? pattern and the second searches in all the files. If you want the output redirected to a file named SEELINE.TXT, change the sole command line to this:

```
for %%x in (%1) do find "%2" %%x > seeline.txt
```

This batch file will not accept multiple words as a search string, for they cannot be assigned to a single replaceable parameter.

Some outputs to the screen cannot be filtered. Here are two examples (notice the single % for a command-line entry):

```
A>for %x in (file?) do type %x ¦ find "Smith"
A>copy file? con ¦ find "Smith"
```

Both will display entire files of all those matching FILE?. The filter is ignored.

The FOR..IN..DO command is often used in complex batch files and is thus explained more fully in Chapter 11.

Practice 8-4

- ☐ 1. Find in ADDRESS (created in the previous session) all the records from one state.
- ☐ 2. Find and sort (by last names in ascending order) the records in another state.
- ☐ 3. Use two search strings to find matching records. (The output should show the normal line and file name.)
- ☐ 4. Do a search but suppress the display of this line:

  ——— ADDRESS
- ☐ 5. Create a batch file named SEEADD.BAT. It should have a fixed search file name, namely ADDRESS. It should allow you to enter a single parameter to search for matching lines.

Inserting Date and Time

- ✎ **Use DATE and TIME with ECHO, FIND, and MORE to insert the date-time in a file.**
- ✎ **Create a batch file that will allow you to add note or diary entries, with the current date and time automatically added after each entry.**

One of the most useful ways of using FIND is inserting date and/or time information in a file. There are occasions when you want to date a file or a piece of information. Entering date and time data can be quite time consuming. Making DOS do it automatically can be time saving.

Adding Date and Time

When you enter the DATE command on the command line, you get this response:

```
A>date
Current date is Mon 06-25-1990
Enter new date (mm-dd-yy):_
```

You can now enter new data at the cursor location. If you combine DATE with ECHO and MORE, the effect is the same as pressing Enter without entering new data:

```
A>echo ¦ more ¦ date
Current date is Mon 06-25-1990
Enter new date (mm-dd-yy):

A>_
```

This mechanism can be used to send the whole message, or just one line, to a file or a device such as a printer.

Now follow these steps and see what happens (make sure there is a practice disk in drive A):

1. Enter the command below:

   ```
   A>echo ¦ more ¦ date > datedata
   ```

   Your disk drive should make a little noise. That means it is saving a file.

2. Display the file on the screen:

   ```
   A>type datedata
   ```

   The file just created should resemble this:

   ```
   Current date is Mon 06-25-1990
   Enter new date (mm-dd-yy):
   ```

3. Use FIND to save only one line:

   ```
   A>echo ¦ more ¦ date ¦ find "C" > datedata
   ```

4. Display the new file:

   ```
   A>type datedata
   Current date is Mon 06-25-1990
   ```

This time only one line is sent to the file. The FIND command searches for a line with an uppercase C and the output is redirected to a file.[2]

Tips & Tricks

Inserting Date-Time without MORE

You can also insert the date-time without using MORE. To provide for a required input, you need to have an existing file. Suppose you already have a file named CR that contains a single carriage return. You can use the batch file below to add the date and time to a file:

```
A>copy con datefile.bat
@echo off
date < cr ¦ find "C" >> notes
time < cr ¦ find "C" >> notes
^Z
```

The CR file will supply the carriage return as required by both DATE and TIME. Every time you run DATEFILE.BAT, the current date and time data will be appended to the file named NOTES.

To make this batch file more versatile, change notes to %1. Then you can use it to insert the date-time in any file, like this:

```
A>datefile report5.bud
```

If you do not want to create a separate CR file, insert the line below after the first line of the batch file:

```
echo/ > cr
```

This line will create a file named CR containing a carriage return. Chapter 11 will explain how it works.

Diary Project

As the final project in this chapter, we are going to create a batch file that will let you make diary (or note) entries and then automatically add date and time data to the end of each entry.

[2] If you would like to use a little trick, the following command will also work:

```
A>echo ¦ more ¦ date ¦ find /v "E"> datedata
```

This eliminates the line with uppercase E and sends the rest to the file.

1. The first step is to create this batch file:[3]

```
A>copy con diary.bat
@echo off
copy diary + con
echo ¦ more ¦ date ¦ find "C" >> diary
echo ¦ more ¦ time ¦ find "C" >> diary
^Z
```

This batch file will append whatever is typed from the keyboard and displayed on the screen to an existing file named DIARY. Before you can use this batch file, you need to create a DIARY file.

2. Create the DIARY file:

```
A>copy con diary
This is my new diary. I will add something every day.

^Z
```

After these two files have been created, you are ready to make a diary entry.

3. Make a diary entry:

```
A>diary
DIARY
CON
Learned some new DOS tricks. Begin to feel confident
about taming this beast.

More ethnic unrest in the USSR. Will there be a Soviet
Union a year from now? I don't think so.

^Z
```

The first two words are automatically displayed by DOS, telling you that what is entered from the console will be appended to the file named DIARY. If you find only CON here, which happens when the DIARY file is nonexistent or inaccessible, your entry will not be saved.

After you press F6 and Enter, what you have typed, plus two more lines, will be saved.

4. Show your diary:

```
A>type diary
This is my new diary. I will add something every day.

Learned some new DOS tricks. Begin to feel confident
about taming this beast.

More ethnic unrest in the USSR. Will there be a Soviet
Union a year from now? I don't think so.

Current date is Mon 06-25-90
Current time is 8:38:21.72p
```

The first line came from the original DIARY file. The two ensuing paragraphs are what you just typed. And the last two lines are automatically added by the batch file.

[3] Version 3.3 users: add a new line "copy diary .+" after "copy diary + con".

Each time you add an entry to your diary by using `DIARY.BAT`, the date-time lines will be added to the end of what you enter, thus meticulously dating (and timing) each entry—all without any effort on your part. With this batch file, you no longer have any excuse for not recording your daily reflections.

As your diary grows, you can use `FIND`, `SORT`, and `MORE` to manage it and look for something quickly. As you are now versed in using these commands, managing such a file should be easy.

Practice 8-5

- ☐ 1. Modify `DIARY.BAT` so that you can use a batch file named `NOTE.BAT` to create a file named `NOTES`. Also arrange for automatic appending of date and time without using the `MORE` command.
- ☐ 2. Use this batch file three times and add a few lines each time.
- ☐ 3. Use `SEELINE.BAT`, created in the previous section, to search for lines in the `NOTES` file matching a search string entered from the keyboard as a parameter.

Drill

_____ 23. The following command is in correct syntax. True or false?

```
A>find "search string" filename
```

_____ 24. The following command will generate an error message. True or false?

```
A>find "string" filename ¦ find "string"
```

_____ 25. The following command will sort found lines. True or false?

```
A>find "string" filename ¦ sort
```

_____ 26. The following command will show the total number of found lines, and display all the lines with sequential numbers. True or false?

```
A>find /c/n "string" filename
```

_____ 27. The following command will find all the files created in January, 1990. True or false?

```
A>dir ¦ find "1-" ¦ find "-90"
```

_____ 28. If you want to search for anything other than "string," the following is a correct command. True or false?

```
A>find /v "string" filename
```

_____ 29. The following command

```
A>find "string1" filename¦find "string2"¦sort > prn
```

will sort and print:

a. lines containing `string1`, plus file name
b. lines containing `string1` OR `string2`, no file name
c. lines containing `string1` AND `string2`, no file name
d. nothing

_____ 30. The following command will print the current date (true or false?):

```
A>echo ¦ more ¦ date ¦ find "C" > prn
```

Summary and Review

Check each item as you review and understand the following highlights of the chapter:

☐ 1. Use > to redirect output to a new device, and < to redirect input from a new device. Use >> to append to a target device. Use ¦ to pipe (connect) two DOS commands.

☐ 2. If a DOS command requires one parameter and sends output to the standard output device, the output can be redirected to another device. For example, instead of sending the output from `DIR` and `TYPE` to the screen, you can redirect it to a new device such as `NUL` or the printer.

☐ 3. If a DOS command requires two parameters, source and target, either one plus the message can be redirected to a new device. For example, you can use `COPY` to send a file from the screen or a disk to `NUL` or the printer. Instead of letting the message (`1 File(s) copied`) go to the screen, you can also redirect it elsewhere.

☐ 4. Some DOS commands such as `FORMAT` and `DISKCOPY` require input from the keyboard before a task is completed. The required input can be redirected from a file on disk. This can save time and automate some jobs.

☐ 5. Use ¦ to pipe together related DOS commands to automate some functions. The output from the first command becomes the input for the second. If the first outputs nothing that the second can use, only the output of the second is displayed.

☐ 6. DOS provides three filter commands (external programs), `MORE`, `SORT`, and `FIND`. These can be used alone or piped to each other or to other commands.

☐ 7. Use MORE to display ASCII text one screen at a time. To do so, redirect a file as input to the command, such as:

```
A>more < filename
```

☐ 8. Use MORE and redirect output to the printer to print text lines entered from the keyboard one line at a time. As soon as Enter is pressed, a line is sent to the printer. The command to do that is:

```
A>more > prn
```

☐ 9. To sort lines entered from the keyboard, run SORT, enter lines to be sorted, and finally press F6 and Enter.

☐ 10. To sort lines of a text file, redirect it as an input to SORT, such as:

```
A>sort < filename
```

☐ 11. Use the /R switch with SORT to sort in descending order, and /+c (where c is a column number) to sort by a certain column. The latter can be used to sort directory entries. For example, to sort by extension, enter this command:

```
A>dir ¦ sort /+10
```

☐ 12. Use FIND to output text lines matching one or more search strings. To use FIND correctly, enter a search string in double quotes followed by a file name, such as:

```
A>find "string" filename
```

☐ 13. FIND can be used with /V to output nonmatching lines, /C to count matching lines, and /N to number and output matching lines.

☐ 14. FIND automatically outputs a one-inch line and a file name. This can be eliminated by entering a second search string without a file name, such as:

```
A>find "string" filename ¦ find "string"
```

☐ 15. Use FIND to insert the date and time in a file—a convenient way to keep track of records. Use the following examples to add the current date and time to a file:

```
A>echo ¦ more ¦ date ¦ find "C" >> filename

A>echo ¦ more ¦ time ¦ find "C" >> filename
```

Review Questions

Write an answer for each question below:

1. What are considered by DOS to be standard input and output devices? What do these devices do?
2. Explain the purpose of each symbol below:

   >, <, >>, ¦

3. How do the following commands differ?

```
A>dir > filename
A>dir < filename
A>dir >> filename
```

4. What happens after each of the following commands?

```
A>copy filename con > nul
A>copy file1 file2 > nul
A>type filename > prn
```

5. What does each of the following commands do?

```
A>echo This is a line. > nul
A>echo This is a line. > prn
A>echo This is a line. > filename
A>echo This is a line. >> filename
```

6. What happens when you run this batch file?

```
A>copy con all.bat
del *.*
y
^Z
```

7. What happens if you enter this command?

```
A>more ¦ dir
```

8. Compare and contrast the following two commands:

```
A>more > prn
A>copy con prn
```

9. What kinds of numbers can be accurately sorted by `SORT`? Why?

10. Explain what each of the following two commands is supposed to do.

```
A>sort > prn
A>sort /+22 filename > prn
```

11. Enter the command to search for all the files created in a specific month of a specific year. Explain the command.

12. Explain what each of the three switches of `FIND` is supposed to do.

13. Enter a command to find matching directory entries and display the items sorted by file size.

14. Enter a command to search in a file for matching lines which are to be sorted and printed without the ————————`FILENAME` line.

15. Explain how this command is supposed to work:

```
A>echo ¦ more ¦ date ¦ find "C" > datedata
```

Configuring the System

Chapter 9

TOPICS:

Reporting Memory Status
The Many Uses of MODE
Speeding Up the Keyboard
Setting Display Modes
Setting Printer Modes
Reporting Device Status
Rerouting Drives and Directories
Configuration Commands
Installable Device Drivers
The DOS Environment
The Environment and Batch Files

COMMANDS COVERED:

MEM, MODE, LASTDRIVE, ASSIGN, JOIN, SUBST, CONFIG.SYS, ANSI.SYS, DEVICE, BREAK, BUFFERS, FILES, SET, VDISK.SYS (RAMDRIVE.SYS), COMSPEC

NEW TERMS AND SYMBOLS:

code page, logical drive, disk buffer, file handle, device drivers, RAM (virtual) disk, DOS environment, environment variable, %variable%

REQUIRED MATERIALS:

1. Two practice disks
2. Printer
3. Access to the following files:
   ANSI.SYS, ASSIGN.COM, JOIN.EXE, MEM.EXE, MODE.COM, SUBST.EXE, VDISK.SYS (RAMDRIVE.SYS)

Quick Start

✎ **To show memory use in detail (version 4 only):**

```
A>mem /debug
```

✎ **To set the keyboard to the highest speed (version 4):**

```
A>mode con rate=32 delay=1
```

✎ **To report device status:**

```
A>mode
```

✎ **To set a dot-matrix printer to 132 columns and 8 lines per inch:**

```
A>mode lpt1 cols=132 lines=8
```

✎ **To assign both A and B to C, and then clear the assignment:**

```
A>assign a=c b=c
A>assign
```

✎ **To substitute a logical drive (Z:) for a real directory, show substitution, and then delete it:**

```
C:\>subst z: c:\ss\central\budget\report
C:\>subst
C:\>subst z: /d
```

✎ **To join a drive to an empty directory, show, and then delete:**

```
A:\>join b: a:\level1\level2
A:\>join
A:\>join b: /d
```

✎ **To increase buffers, file handles, and install a RAM disk and ANSI.SYS, include these in CONFIG.SYS:**

```
buffers=20
files=20
device=vdisk.sys      (ramdrive.sys for MS-DOS)
device=ansi.sys
```

✎ **To expand the environment to 512 bytes, add this to CONFIG.SYS:**

```
shell=c:\command.com /p /e:512
```

✎ **To use a RAM disk for copying in a one-floppy system:**
Use the RAMCOPY.BAT batch file

✎ **To add path items to an existing path list:**
Use the ADDPATH.BAT batch file.

✎ **To quickly set a prompt style:**
Use the SETPROM.BAT batch file.

This chapter deals mostly with defining and modifying your hardware parts. A host of DOS commands and special files are available to control their performance or report their current status. Most of these are covered in this chapter.

In the order listed below, you will learn to:

- ✎ **Use MEM to report memory use**
- ✎ **Use MODE to report device status, and control monitor, printer, and keyboard**
- ✎ **Use ASSIGN, JOIN, and SUBST to reroute drives and directories**
- ✎ **Use CONFIG.SYS to set BREAK, BUFFERS, FILES, and LASTDRIVE values, and install device drivers**
- ✎ **Become familiar with the DOS environment and maneuver it with batch files**

Reporting Memory Status

✎ **Use MEM to report memory use; use it with /PROGRAM or /DEBUG switch to provide a detailed report.**

Version Variances: The MEM command discussed in this section is not available for versions before 4.0.

If you wonder what is in the invisible world of computer memory, you can use a command available in version 4 to tell you. The MEM.EXE program can give you a simplified report if no parameter is entered, such as:

```
A>mem

  655360 bytes total memory
  655360 bytes available
  517152 largest executable program size

  393216 bytes total extended memory
  393216 bytes available extended memory
```

The above numbers are based on a one-megabyte system. They show 640KB, which DOS can use and 384KB extended memory, which DOS cannot use; the latter can be used as a RAM disk as explained below.

The third number in the above display shows the amount of memory for your use. It is the same as the free memory reported by CHKDSK. It is the leftover after deducting from 640KB the amount used by programs already loaded, including DOS itself.

If you want to see a detailed report of the programs in memory and their locations (displayed in hex numbers), use the /PROGRAM switch:

```
A>mem /program
```

If you have installed PRINT and APPEND, for example, their names and memory locations will also show up in the display. If you want more details (including input-output devices), try the /DEBUG switch:

```
A>mem /debug
```

The detailed information is useful to programmers, but not to most ordinary DOS users. If you are not a programmer, you need only know what programs are already loaded and whether there is enough memory to load another program.

The Many Uses of MODE

✎ **MODE's various uses are summarized.**

MODE is a jack-of-all-trades command that can be used to control many devices connected to the computer. You can use it to control the following:

✎ **Keyboard speed**

✎ **Parallel adapter, usually connected to a printer**

✎ **Serial adapter, connected to a printer, mouse, or modem**

✎ **Video display modes**

✎ **Report status of a device**

Other peripheral devices connected through a parallel port, serial port, keyboard, and monitor jacks can all be controlled by using MODE. These devices can also be instructed to accept and output certain code pages (foreign languages). You are not likely to need these. We will concentrate on using American English.

Speeding Up the Keyboard

✎ **Use MODE CON followed by specifics to set keyboard speed.**

Your keyboard normally operates at a snail's pace. To test how slowly it repeats itself, hold down a letter key, "x" for example. Then hold down the Backspace key to delete them—ever so slowly.

Version Variances: Users of version 4 can get DOS relief from this torturously slow pace. Users of earlier versions will have to find relief somewhere else—from third-party programs such as Cruise Control.

If you have MODE.COM in drive A or have provided a path to the program, you can improve your keyboard's speed by entering this command:

```
A>mode con rate=32 delay=1
```

Do not forget the CON. Here you are telling DOS to set a console speed.

Now, repeat pressing a letter key and then Backspace to see the difference. The cursor now travels more than twice the earlier speed.

You must enter both values together; otherwise, this error message appears:

```
RATE and DELAY must be specified together
```

A keyboard normally operates at the rate of repeating 20 times per second with a delay of 0.5 second. That is the same as entering this command:

```
A>mode con rate=20 delay=2
```

Each delay is 0.25 second. You are allowed to enter delay values as short as 1 (0.25 second) and as long as 4 (1 second). Rate is the speed a keystroke is repeated every second. You can go from as low as 1 to as high as 32. If you want your keyboard automatically set at a higher speed every time you begin to use it, you should include a proper command in your AUTOEXEC.BAT file.

MODE CON may not work with your keyboard if your PC's BIOS does not support a higher keyboard speed. It does not work on an IBM keyboard connected to an original IBM PC.

Tips & Tricks

Speedy Cursor

Some commercial software packages will allow you to dramatically speed up your keyboard. The most famous is Cruise Control.

I use a package called Repeat Performance provided by WordPerfect Corporation. I use it to set a keystroke repeat rate of 350 per second with a minimum delay speed of 0.25 second. This speed is about 10 times the highest rate you can set with MODE.

The higher cursor speed can increase productivity. While doing word processing, I can move the cursor much more quickly. On the DOS command line, pressing a key without the slightest delay displays a whole line of the same letter. Pressing Backspace or the left arrow key erases them just as fast.

Setting Display Modes

✎ **Use MODE to set mono or color 40/80 modes on a CGA system. Use MODE in version 4 to set EGA/VGA to 25, 43, or 50 lines.**

The normal monitor displays text 80 columns by 25 lines. If you have a special graphics card (adapter) installed in your computer, such as a CGA (Color Graphics Adapter), EGA (Enhanced Graphics Adapter), or VGA (Video Graphics Array), you can change these numbers. If you have a CGA card and an RGB (Red Green Blue) color monitor, you can use MODE to set these values:

MONO	Monochrome (no color) display
BW40	Black and white, 40 columns
BW80	Black and white, 80 columns
CO40	Color, 40 columns
CO80	Color, 80 columns

To disable colors, enter this:

```
A>mode mono
```

This is the same as BW80; both disable colors. It is the only option for a monochrome system.

This command changes to 40-column color display:

```
A>mode co40
```

If you have a monochrome monitor with an MDA (Monochrome Display Adapter) or Hercules (monochrome graphics) card, any option except MONO will lead to an error message, such as:

```
A>mode bw80
Function not supported on this computer
```

If you have a CGA system and the screen display is not centered, you can shift it left or right, such as:

```
A>mode co40, r, t
```

This shifts to the right by one column and tests the new display. Or you can try this:

```
A>mode co80, l, t
```

This shifts to the left by two columns (each shift is by two when in the 80-column mode) and tests the result.

You cannot shift the screen with a monochrome, EGA, or VGA system. If you try, you will get a message like this:

```
Unable to shift screen left
```

Version Variances: Starting with version 4.0, you can not only switch between 40 and 80 columns, but also change from 25 to 43 or 50 lines of text per screen with an EGA or VGA system. You need to install `ANSI.SYS` with the `/L` switch as explained below. After that, you can try this command:

```
A>mode con cols=80 lines=43
```

If you do not need to reset the column number, you can just enter:

```
A>mode con lines=43
```

If the display looks unattractive, you can change it to 25 lines or take out the `/L` switch of `ANSI.SYS` from your `CONFIG.SYS` file.

Setting Printer Modes

✎ Use MODE to set printing at 6/8 lines per inch and/or 80/132 columns.

You can use `MODE` to control your printer to print the number of characters per line and the number of lines per inch. The syntax goes like this:

```
A>mode lpt1 cols=# lines=#
```

The `cols=` and `lines=` notations are optional; if they are not used, you need to put a comma between them. Here you are supposed to supply 80 or 132 in place of the first #, and 6 or 8 in place of the second, such as:

```
A>mode lpt1 cols=132 lines=8 retry=r
LPT1: not rerouted
LPT1: set for 132
Printer lines per inch set
Resident portion of MODE loaded
Infinite retry on parallel printer time-out
```

Your printer should be turned on when you enter this command. If not, you will get the `Printer error` message and the printer will not be affected by the `MODE` command.

Notice the use of `LPT1`. You can put a colon at the end if you wish; it will make no difference. Do not use `PRN`. If you do, you will get an error message like this:

```
A>mode prn cols=132
Invalid parameter - 132
```

The `RETRY` parameter is optional. It is to respond to a busy or not-ready status of the parallel port. You can enter one of these values: B (busy), E (error), N (none), or R (retry). If no parameter is entered, N is the default.

Version Variances: In versions before 4.0, you are to enter column and line numbers separated with a comma, such as:

```
A>mode lpt1 132, 8, p
```

P is optional. It is the same as RETRY=R in version 4. It will continuously try to send output to the parallel port even though it may be busy or not ready. Press Ctrl-C if you want to regain control of the computer. Specifying the P (or R in version 4) option will cause MODE . COM to stay resident until power is off.

Now follow these steps to see how MODE affects your dot-matrix printer connected to the first parallel port (if MODE . COM is not in drive A, provide a path to it):

1. Change to the 132-column mode:

   ```
   A>mode lpt1 132
   ```

   If your printer is on, the message below appears:

   ```
   A>mode lpt1 132
   LPT1: not rerouted
   LPT1: set for 132
   No retry on parallel printer time-out
   ```

2. Send a line to the printer:

   ```
   A>echo This should be printed in tiny letters > prn
   ```

   The characters should be printed at the rate of 16.5 characters per inch.

3. Restore the 80-column mode:

   ```
   A>mode lpt1 80
   LPT1: not rerouted
   LPT1: set for 80
   No retry on parallel printer time-out
   ```

4. Print another line:

   ```
   A>echo This is the regular print. > prn
   ```

   The line should be printed at the rate of 10 characters per inch.

The above steps should work with modern dot-matrix printers. Some older models may not work. A daisy-wheel printer is not likely to obey these commands.

Reporting Device Status

✎ **Use MODE to report the status for all or a specific device.**

We have used MODE to control various DOS devices. We can also use it to report (display on the screen) the available devices and the status of each. To do that, simply enter the command without any parameter:

```
A>mode
```

The screen displays a report of all the devices:

```
Status for device LPT1:
-----------------------
LPT1: not rerouted
RETRY=NONE
Code page operation not supported on this device

Status for device LPT2:
-----------------------
LPT2: not rerouted
RETRY=NONE
Code page operation not supported on this device

Status for device LPT3:
-----------------------
LPT3: not rerouted

Status for device CON:
----------------------
COLUMNS=80
LINES=25
Code page operation not supported on this device
```

If you want to see the status of a single device, place the device name after MODE, such as:

```
A>mode con
```

With this command, only the last part of the above report will be displayed.

The above display shows no printer rerouted (redirected). You can redirect LPT1 to the first serial port this way:

```
A>mode lpt1=com1

Resident portion of MODE loaded
LPT1: rerouted to COM1:
```

You can now report the status of LPT1:

```
A>mode lpt1 /status

Status for device LPT1:
-----------------------
LPT1: rerouted to COM1:
RETRY=NONE
Code page operation not supported on this device
```

The /STATUS switch is used to report code page information. It can be shortened to /STA. Without this switch the last line above will not be shown.

Code page is Microsoft's fancy term for a number designated for one of the supported foreign languages. You can use MODE to set a value so that a printer or monitor can output symbols unique to that particular language. That should not concern most American users.

In using a serial port, you can set various values, such as:

```
A>mode com1 baud=2400 stop=1 data=8 parity=none
COM1: 2400,n,8,1,-
```

You can set these values for a modem or printer connected to the first serial port. Your modem or serial printer should provide instructions as to what values to set. In the case of a modem, the baud rate can be set by the communications program you use.

Practice 9-1

- ☐ 1. Use MEM to show the most detailed information about your computer's memory use.
- ☐ 2. Use MODE to set your keyboard to the highest possible speed.
- ☐ 3. If you have a color-graphics system, use MODE to set:
  a. 40-column color
  b. 80-column black and white
  c. 80-column color
- ☐ 4. If you have a dot-matrix printer, do the following:
  a. Set to 132 columns and use ECHO to print a line
  b. Set to 80 columns and use ECHO to print a line
- ☐ 5. Use MODE to report the status of the monitor and printer.

Drill

_____ 1. Entering the MEM command without a switch can tell you what programs are loaded in the memory. True or false?

_____ 2. Both MEM and CHKDSK can show the amount of available memory. True or false?

_____ 3. If the command A>mode is entered, the result is to:
  a. report the status of all devices
  b. set all devices to default values
  c. do both a and b
  d. generate an error message due to illegal use

_____ 4. In setting keyboard speed, you can set RATE and DELAY separately. True or false?

_____ 5. The command `A>mode lpt1 120` will:
   a. set 120-column printing
   b. set 132-column printing
   c. do nothing
   d. generate an error message

_____ 6. You can use `MODE` to shift screen display on any kind of monitor. True or false?

_____ 7. The command `A>mode lpt1:=com1:` will allow you to use a printer connected to the first serial port. True or false?

_____ 8. The command `A>mode con` will:
   a. display options to let you set a monitor mode
   b. report monitor status
   c. do nothing
   d. generate an error message

Rerouting Drives and Directories

✎ **Use `ASSIGN` to assign an existing drive to another.**

✎ **Use `SUBST` to substitute a real or logical drive for an existing directory, which need not be empty.**

✎ **Use `JOIN` to join an empty directory to a real drive.**

Just as you use `PATH` or `APPEND` to make DOS search for a file not in the current directory, you can use `ASSIGN`, `JOIN`, and `SUBST` to make DOS treat a drive or directory in a different way. You can use `ASSIGN` to assign one drive to another, `JOIN` to join one drive to another, and `SUBST` to substitute one drive for another.

Keep in mind that all three commands are external. You have to put them in drive A before doing the following. Or you must provide a path to where you store these commands.

Assigning Drives

If both drives A and B have practice disks and doors closed, you are ready to assign drives:

1. Assign A to B:
   ```
   A>assign a=b
   ```

   Think of the above as "assign A to B" or assigning the first drive to the second. You can put a space before or after =. Version 4 will accept a colon (:) after a drive letter, but earlier versions will not permit that.

2. Show the identical result with any command below:

```
A>dir
A>dir a:
A>dir b:
```

Any of these commands will take you to drive B. Even though the default drive is still A, the first command will display the directory of drive B. After this assignment, anytime you want to go to A, DOS will take you to B.

3. Clear the assignment:

```
A>assign
```

If you started out with ASSIGN.COM stored on the disk in drive A, you will get an error message because DOS can no longer access drive A. In that case, temporarily swap the disks between A and B and reenter the above command.

4. Show the result:

```
A>dir
```

You are now back to the original arrangement. Entering the ASSIGN command alone clears any previous assignment, if any. You can now return the disks to their original drives.

You can do multiple assignments in one command, such as:

```
A>assign a=c b=c
```

This assigns both A and B to C. If an application program demands access to either drive A or B, you can copy the required files to C and then enter the above command. Whenever the program needs to go to A or B to access a certain file, your computer will be tricked to go to the hard disk.

Substituting Drives

Next, we are going to create more logical drives and substitute one of them for a real drive. A **logical drive** is one that is recognized by DOS as a legitimate drive. If your hard disk is divided into two logical drives, DOS will recognize them as two drives even though there is only one physical drive.

A logical drive may be just a letter without any physical existence. You can, however, substitute it for a real thing, as we intend to do below. To do the following steps, you need to include this line in your CONFIG.SYS file:

```
lastdrive=z
```

This command tells DOS to use all the available drive letters, up to Z.

Find out whether your CONFIG.SYS includes this line by using a command like this one:

```
A>type c:\config.sys
```

If the line is not there, enter a command like this:

```
A>copy c:\config.sys + con
lastdrive=z
^Z
```

If it is not possible to change the CONFIG.SYS file on the hard disk, create your own CONFIG.SYS file and save it to a boot disk. Reboot with this disk after saving this file:

```
A>copy con config.sys
lastdrive=z
^Z
```

This adds the arrangement necessary for our steps below. After the line is saved, reboot your computer and follow these steps:

1. Set a new prompt:

   ```
   A>prompt $p$g
   A:\>_
   ```

2. Create a new directory:

   ```
   A:\>md level1
   ```

3. Go to the new directory:

   ```
   A:\>cd level1
   A:\LEVEL1>_
   ```

4. Create a subdirectory:

   ```
   A:\LEVEL1>md level2
   ```

5. Change to the new subdirectory:

   ```
   A:\LEVEL1>cd level2
   A:\LEVEL1\LEVEL2>_
   ```

6. Go to the root directory:

   ```
   A:\LEVEL1\LEVEL2>cd\
   A:\>_
   ```

7. Substitute drive Z for the subdirectory:

   ```
   A:\>subst z: a:\level1\level2
   ```

8. Show the directory of A:\LEVEL1\LEVEL2:

   ```
   A:\>dir z:
   ```

   The directory entries of A:\LEVEL1\LEVEL2 are displayed. This can save you the trouble of typing the long path.

9. Show the SUBST list:

   ```
   A:\>subst
   Z: => A:\LEVEL1\LEVEL2
   ```

   This shows that the Z drive is assigned the value pointed to by the arrow.

10. Delete the SUBST list:

    ```
    A:\>subst z: /d
    ```

Before you can substitute Z for another directory, you must delete its assigned value as shown above. Otherwise, this error message appears:

```
Invalid parameter - z:
```

The SUBST command can be a timesaver if you use a hard disk and a complex directory structure. Suppose your report files are located in this directory:

```
C:\SS\CENTRAL\BUDGET\REPORT
```

Suppose you now enter this command to substitute R for this long directory path, like this:

```
C:\>subst r: c:\ss\central\budget\report
```

Any time you want to do something with regard to the REPORT directory, just use R: to replace the long path. For example, if you are in another directory, you can quickly go to that directory by entering its designated drive letter:

```
C:\WP\LETTER>r:
R:\>_
```

Although the DOS prompt shows the substituted letter as the current directory, you are actually in the REPORT directory.

Note: In a SUBST operation, as shown above, the directory to be substituted with a drive letter does not have to be empty. In a JOIN operation, discussed below, however, the directory to be joined to a drive must be empty.

Joining Drives

You can use the JOIN command to join two drives together. This can physically expand a disk or directory. To demonstrate how to use this command, we are going to use the empty directory created in the preceding steps and join drive B to it. If you have not changed the previous arrangement, you are ready to go:

1. Join drive B to A:\LEVEL1\LEVEL2:

   ```
   A:\>join b: a:\level1\level2
   ```

   The target drive's directory must be empty; otherwise, this error message appears:

   ```
   Directory not empty - a:\level1\level2
   ```

2. Go to the subdirectory:

   ```
   A:\>cd\level1\level2
   A:\LEVEL1\LEVEL2>_
   ```

3. Show the directory:

   ```
   A:\LEVEL1\LEVEL2>dir
   ```

   Instead of the current directory of drive A, the entries in drive B are read and displayed.

4. Show drive A's directory:

```
A:\LEVEL1\LEVEL2>dir a:
Invalid drive specification
```

DOS no longer recognizes the joined directory in drive A; it will, however, recognize other directories.

5. Create and save a file:

```
A:\LEVEL1\LEVEL2>copy con test
Where does this file go?
^Z
```

The file is saved to drive B.

6. Delete the file:

```
A:\LEVEL1\LEVEL2>del test
```

The file is deleted from drive B.

7. Show the joined drives:

```
A:\LEVEL1\LEVEL2>join
B: => A:\LEVEL1\LEVEL2
```

If you get the `Bad command or file name` error message, it is because DOS can no longer access `JOIN.EXE`. One simple remedy is to provide a path, such as:

```
A:\LEVEL1\LEVEL2>path a:\
```

This will guide DOS to search for the command in `A:\`. If you enter `JOIN` now, you should have no more trouble.

8. Detach the joined drives:

```
A:\LEVEL1\LEVEL2>join a: /d
```

The two drives are now disconnected. If you now enter the JOIN command alone, nothing is shown. DOS also begins to recognize the previously joined directory in drive A as a separate entity.

9. Remove all the directories created by the above steps.

Joining drives allows you to expand one disk to another. If one disk is about to run out of space, you can create an empty subdirectory and join another drive to it. Under some circumstances, for example in some application programs, this may enable you to run your program smoothly.

Tips & Tricks

The Hazards of Rerouting Drives

The three commands used to reroute drives should not be used together. If you join two drives together and then substitute one of them for another, DOS could be really confused and you may get unexpected results.

If a drive is rerouted, some commands will not work and others may produce surprising outcomes. When a drive is rerouted, do not use these commands:

```
BACKUP, DISKCOPY, DISKCOMP, FDISK, FORMAT, PRINT,
RESTORE
```

Before you use any of these commands, clear any rerouting.

Practice 9-2

- ☐ 1. Assign drive B to drive A and use `DIR` to show the result.
- ☐ 2. Delete the above assignment and use `DIR` to prove it.
- ☐ 3. Add the following line in your `CONFIG.SYS` and reboot (ignore this if you have already done it):

```
lastdrive=z
```

- ☐ 4. Create an empty directory named `A:\XYZ` and substitute drive X for it. Delete the substitution.
- ☐ 5. Join the above empty directory to B. Delete the joining.
- ☐ 6. Delete the substitution and joining.

Drill

_____ 9. After assigning drive A to drive B (`a=b`), you can no longer access A before another reassignment or the assignment is deleted. True or false?

_____ 10. The second command below will produce:

```
A>subst k: a:\nano
A>subst
```

a. A:\NANO => K:
b. K: => A:\NANO
c. A: => K:\NANO
d. an invalid parameter

_____ 11. In the above arrangement, A:\NANO must be empty. True or false?

_____ 12. After the steps in question 10, the command A>dir k: will show the directory entries of A:\NANO. True or false?

_____ 13. After the steps in question 10, it is legal to do the following:
A>assign a=k. True or false?

_____ 14. If SUBST is replaced with JOIN in question 10, A:\NANO must be an empty directory. True or false?

_____ 15. Entering ASSIGN without any parameter clears any existing assignment. True or false?

_____ 16. Entering JOIN without any parameter clears (disconnects) any joined drives. True or false?

_____ 17. After the following three commands, the screen will show the directory entries of:

```
A>subst z: b:\bat
A>assign a=z
A>dir a:
```

a. A:\
b. Z:\
c. B:\
d. B:\BAT

Configuration Commands

- **Put configuration commands in a CONFIG.SYS file stored in the root directory of a system disk to change default values in the system.**
- **Increase BUFFERS and FILES numbers to enhance your computer's capacity.**

CONFIG.SYS is a file created by you to define how your system is supposed to work. It can contain configuration commands that enable your hardware and software to perform specific functions, such as handling more files, which they otherwise cannot do.

CONFIG.SYS

You can put more than a dozen commands or device drivers in a CONFIG.SYS file. You are, however, not likely to need more than the following:

```
BREAK
BUFFERS
FILES
DEVICE=ANSI.SYS
DEVICE=VDISK.SYS
```

A CONFIG.SYS file must be in the root directory of a system (boot) disk. After the hidden files on the system disk are run, CONFIG.SYS takes its turn and finally AUTOEXEC.BAT runs. To create a CONFIG.SYS file, you may do something like that below:

```
A>copy con config.sys
break on
buffers=20
files=20
device=ansi.sys /x
device=vdisk.sys 120
^Z
```

BREAK

The BREAK command can be put in CONFIG.SYS or entered on the command line. Follow these steps to see how it works:

1. Check to see whether BREAK is on or off:

   ```
   A>break
   BREAK is off
   ```

   This is the default state. If you do nothing, BREAK is off.

2. Turn on BREAK:

   ```
   A>break on
   ```

3. Check the new state:

   ```
   A>break
   BREAK is on
   ```

The state of BREAK determines when DOS is to check for Ctrl-Break and interrupt an activity. When BREAK is off, DOS checks for Ctrl-Break only when working with the standard input (keyboard), standard output (monitor), serial and parallel ports.

When BREAK is turned on, DOS checks for Ctrl-Break while performing any task. This allows you to interrupt a program running in RAM or a disk-accessing activity. If BREAK is off and something goes wrong, such as a program stuck in a perpetual loop, you cannot interrupt it.

To compensate for more frequent checking when BREAK is on, DOS will naturally slow down. It is thus advisable to leave BREAK off normally. When you do a great deal of program testing or disk accessing, you can temporarily turn it on.

To demonstrate how BREAK can affect the computer, first create this batch file:

```
A>copy con brktest.bat
@echo off
:LOOP
set a=aardvark
goto LOOP
^Z
```

Run the batch file after turning on BREAK. Terminate the batch file with Ctrl-Break.

Next, turn off BREAK and run the batch file. This time the computer is stuck in a perpetual loop and you cannot break out of it. You must now reboot the computer.

If the first line of the batch file is deleted, you can terminate the batch file regardless of the status of BREAK. The reason is that the batch file writes to the monitor, which DOS can intercept.

The SET command is discussed below. The looping mechanism will be explained in Chapter 11.

BUFFERS

BUFFERS and FILES are strictly configuration commands and must be entered from a CONFIG.SYS file. DOS cannot recognize them when they are entered from the command line.

BUFFERS determines the number of disk buffers. A **disk buffer** is a memory area reserved to store disk information for quick action and reduced access to the physical disk. Each buffer takes up 528 bytes of memory, which is slightly bigger than a sector.

You can set BUFFERS to a value of 1–99. If you do nothing, DOS assumes a default value based on the available hardware as shown in Table 9.1.

Table 9.1 Disk Buffers

Buffers	Hardware Configuration
2	128K RAM, 360 drive
3	higher than 360 drive
5	more than 128K RAM
10	more than 256K RAM
15	more than 512K RAM

For most occasions, 20–30 buffers are considered optimal. If you do frequent random (rather than sequential) disk reading and writing, you may speed up your work by having more buffers. However, more than 30 buffers will generally slow down your computer.

Version Variances: In version 4, you can add a look-ahead number and use expanded memory for buffers with a command in this syntax:

```
buffers=number, ahead /x
```

The number of buffers in the regular memory is still limited to 99. But if you use the `/X` switch to put them in expanded memory, you can have up to 10000 buffers.

A look-ahead number can be 1–8, with the default value of 1. This tells DOS the number of buffers to look ahead when reading data. The more you specify, the more DOS will speed up sequential (not random) reading.

FILES

`FILES` specifies the number of files that can be opened at one time. If you do nothing, DOS can handle up to 8 files simultaneously. You can specify up to 255.

To enable DOS to handle a file, each file is given a **file handle**, which is a code number. Each file handle takes up 48 bytes of memory.

An application program requiring more than the default of 8 file handles will tell you how many to specify. If you encounter an error message like `Run out of file handles`, you can no longer access another file. The simple remedy is to increase the value of `FILES`.

Installable Device Drivers

- **Include `VDISK.SYS` (`RAMDRIVE.SYS`), and `ANSI.SYS` in your `CONFIG.SYS` file to create a RAM disk and install the ANSI driver.**
- **Use a RAM disk to store information for quick access by DOS or to do copying in a one-floppy system.**

In the `CONFIG.SYS` file discussed in the previous section, we included a couple of **device drivers**, special files used to drive specific devices. There are more than half a dozen such drivers, some of which can be used to redefine the keyboard, monitor, or printer to accommodate foreign languages. Most of these should not concern you, except two—`ANSI.SYS` and `VDISK.SYS`.

`ANSI.SYS` is used to enhance the keyboard and monitor. Once installed, it can be used to define many keys on your keyboard and set screen colors and cursor positions. To install this driver, you must include a line like the one below in `CONFIG.SYS`:

```
device=c:\dos\ansi.sys /x
```

This command goes to `C:\DOS` to install `ANSI.SYS`; this file must be stored in the designated directory. If your arrangement is different, you must direct DOS to the directory storing the `ANSI.SYS` file.

Version Variances: If you use version 4, you can specify two of these three available options:

`/K`	`Kill` (ignore) the extra keys on the enhanced keyboard
`/L`	Long display, 43 or 50 lines of text for EGA or VGA; use `MODE` to set a number of text lines
`/X`	Extra keys on the enhanced keyboard definable

If you have an EGA or VGA system, the `/L` switch will allow you to display more text lines. Once `ANSI.SYS` is installed with this switch, you can use `MODE` to set the number of lines as explained earlier in this chapter.

The enhanced keyboard with 101 keys can be greatly enhanced with the `/X` switch. With this switch, you can define or redefine the F11 and F12 function keys and the extra cursor keypad in the middle. If your application program does not allow the use of the extra keys, you may need to use the `/K` switch to make peace with it.

`ANSI.SYS` can vastly increase the capabilities of DOS. Anybody who wants to master DOS should learn to take advantage of it. Chapter 12 is exclusively devoted to it. You can better utilize it after you learn to use EDLIN (Chapter 10) and to write programlike batch files (Chapter 11).

Creating a RAM Disk

`VDISK.SYS` (known as `RAMDRIVE.SYS` in MS-DOS) is used to create a **RAM disk** (or virtual disk). A RAM disk is a separate area of memory that can act as a disk drive. You can copy files to and from it. You can temporarily store files there; the files disappear when power is off. You can store there accessory files needed by an application program, such as WordStar overlay files; this can cut down access to disk and greatly speed up action.

You can maneuver a RAM disk just like a regular disk drive. You can create multiple RAM disks. In each disk, you can create directories and subdirectories. You can even use `CHKDSK` on it. I use my AT's extended memory to create a 384KB disk to store WordPerfect overflow files with this command in `CONFIG.SYS`:

```
device=c:\dos\ramdrive.sys 384 /e
```

When I check this simulated disk, this is the outcome:

```
E>chkdsk f:

Volume MS-RAMDRIVE created 06-09-1985 12:00a

    389120 bytes total disk space
      4096 bytes in 5 user files
    385024 bytes available on disk

       512 bytes in each allocation unit
       760 total allocation units on disk
       752 available allocation units on disk

    655360 total bytes memory
    513296 bytes free
```

To create a RAM disk, you need to include a line like the one below in your CONFIG.SYS file:

```
device=c:\dos\vdisk.sys 384 256 64 /e
```

This assumes that VDISK.SYS is stored in the C:\DOS directory. If your arrangement is different, change it accordingly.

Notice there are three numbers and a switch. Actually they are all optional. If you do not specify them, certain default values apply.

The first number specifies the size of the RAM disk. If not specified, 64KB is the presumed size. If the value you specify exceeds available memory, the default value will be used.

The second number is for sector size. If omitted, 128 byte is used. You can specify 128, 256, or 512. If you store small files in the RAM disk, specify a smaller number. Otherwise, let DOS determine the number for you.

The third number is for the number of directory entries that can be stored on the RAM disk. If omitted, 64 is the default. The range can be from 2 to 512.

The /E switch is for extended memory. Most AT (286 and 386) or PS/2 computers are equipped with 1MB RAM. Since DOS can use only 640KB, you can use the remaining 384KB for a RAM disk. The /E switch tells DOS to use the otherwise inaccessible memory.

Version Variances: Instead of /E, you can use /A in version 4. That tells DOS to use expanded memory conforming to the LIM (Lotus-Intel-Microsoft) specifications. If you have a choice, use /E rather than /A because most application programs can use expanded, but not extended, memory. It is thus a waste of your resources to use expanded memory for a RAM disk.

You can set up two or more RAM disks. Just enter in CONFIG.SYS as many lines for as many disks as you wish to create, such as:

```
device=c:\dos\vdisk.sys
device=c:\dos\vdisk.sys
```

This will set up two RAM disks using default values.

Using a RAM Disk

If you have one floppy drive and a hard disk—which is a common configuration for many people—you are likely to encounter a great deal of inconvenience when you copy files from one floppy disk to another. If you happen to have unused and unusable RAM, you can create a RAM disk and copy files between it and the floppy drive.

After a RAM disk is created, you can create a batch file to make the job of copying much easier. The batch file below will copy files from A to F, pause for you to swap disks, then copy the same files from F to A, and finally delete the same files in F:

```
A>copy con ramcopy.bat
@echo off
rem RAMCOPY.BAT--copy from A to RAM disk and back
if .==%1. goto END
echo Insert source disk in A and
pause
copy a:%1 f:
echo Copy from A to F is complete.
echo Insert target disk in A and
pause
copy f:%1 a:
del f:%1
:END
```

Change F to your RAM disk's specific letter if it is different. This letter is determined by DOS when creating a RAM disk. It uses the letter next to the letter used for the last installed drive.

With this batch file, you can specify one file or a series of matching files to copy. The following shows a sequence of events for copying matching files:

```
C:\BAT>ramcopy file?
Insert source disk in A and
Press any key to continue . . .

A:FILE1
A:FILE2
        2 File(s) copied
Copy from A to F is complete.
Insert target disk in A and
Press any key to continue . . .

F:FILE1
F:FILE2
        2 File(s) copied
C:\BAT>
```

If no parameter follows the batch file name, batch execution branches to the `END` label and nothing is copied.

This batch file contains label, parameter, parameter testing, and several batch commands. All will be explained in Chapter 11.

Practice 9-3

- ☐ 1. Find out whether `BREAK` is on or off. Change it to the opposite state and then return it to the original.
- ☐ 2. Create a RAM disk with 64KB size and 128 byte sector.
- ☐ 3. Install `ANSI.SYS` with default values.
- ☐ 4. Use `RAMCOPY.BAT` to copy a file from drive A to the RAM disk and back.

Drill

_____ 18. The status of BREAK is off by default. True or false?

_____ 19. When BREAK is off, you can never press Ctrl-C to terminate a batch file. True or false?

_____ 20. CONFIG.SYS can be stored in any directory and DOS will have no trouble finding and executing it. True or false?

_____ 21. Disk buffers do not use memory and so you have nothing to lose by using a larger number. True or false?

_____ 22. It is preferable to use extended rather than expanded memory for a RAM disk. True or false?

_____ 23. DOS versions 3.3 and 4.xx can both use ANSI.SYS to define the extra keys on the enhanced keyboard. True or false?

_____ 24. If you do not specify any parameter with VDISK.SYS (RAMDRIVE.SYS), DOS will create a RAM disk with 64KB. True or false?

The DOS Environment

- **Use the SHELL command to expand the environment.**
- **Use the environment to manage APPEND, PATH, and PROMPT.**

The area of memory that DOS sets aside to keep track of path, prompt, and other information is called the environment, or **DOS environment**. You can use the SET command to see what is stored in the environment:

```
A>set
COMSPEC=C:\COMMAND.COM
PATH=D:\C\BIN;D:\C;
LIB=D:\C\LIB;
INCLUDE=D:\C\INCLUDE;
INIT=D:\C\SOURCE\ME\INI;
```

These strings are put in the environment by the AUTOEXEC.BAT file after DOS is booted. The first line tells DOS to load COMMAND.COM from C:\ when necessary.[1] The remaining lines are to guide the QuickC compiler to compile and link a C program code. Your application program may require a different set of strings.

PATH, PROMPT, and APPEND (if installed with the /E switch) are entered into the environment whenever such a command is entered on the command line or from a batch file. Try these steps to see how you change the environment:

Tips & Tricks

Expanding the Environment

The environment space is by default limited to 160 bytes. DOS will expand this space as needed. However, DOS cannot expand in two situations: when a batch file is being executed and after a memory-resident program occupies a memory area, thus blocking expansion to a contiguous area. You may then get this message:

```
Out of environment space
```

That means there is no more space available to store your addition. One remedy is to reserve more memory for the environment with a command like this added to CONFIG.SYS:

```
shell=c:\command.com /p /e:512
```

This specifies the environment size of 512 bytes and instructs DOS to keep COMMAND.COM permanently in the memory (instead of exiting right after loading). The /P switch is necessary, and the /E switch lets you specify 160 to 32767 bytes.

[1] When a large application program is loaded, a portion of COMMAND.COM will be pushed out of RAM. If you need to use a DOS command that is pushed out, DOS must know where to reload COMMAND.COM. In this situation, you need to have a line like the one below in your AUTOEXEC.BAT file to direct DOS to the right place:

```
comspec=c:\command.com
```

1. Enter a path:

```
A>set path=b:\bat
```

You can omit SET when entering PATH, PROMPT, or APPEND into the environment. If you want to create your own environment variable, however, you must use SET.

2. See the environment:

```
A>set
. . . .
PATH=b:\bat
```

The earlier path list, if any, is replaced by the new one. DOS automatically displays the string before = in uppercase.

3. Display path list:

```
A>path
PATH=b:\bat
```

The response is taken right out of the environment.

4. Remove the path from the environment with either command below:

```
A>set path=
A>path;
```

5. See the environment again:

```
A>set
```

This time there is no more PATH command in the environment. If you enter the PATH command alone, you will get the No Path message.

In all the above steps, PATH could be replaced with APPEND with identical results. PROMPT also works in a similar manner. PATH, APPEND, and PROMPT were explained in Chapter 5. PROMPT will be covered more thoroughly in Chapter 12.

The Environment and Batch Files

✎ **Use batch files to maneuver environment variables.**

You can enter your own **environment variables** (also known as named parameters) by using the SET command, such as:

```
A>set password=open sesame
```

After entering this command you can use the SET command alone to see the environment having this string:

```
PASSWORD=open sesame
```

DOS automatically converts the variable, which is the string before =, to uppercase and keeps the value, which is the string after =, the way you type it.

Tips & Tricks

Environmental Cases

Case and spacing may play an important role in an environment variable. Consider the following examples:

```
A>set password=open sesame
A>set password = open sesame
A>set password=Open Sesame
```

They are all different from one another. They will respectively enter the following into the environment:

```
PASSWORD=open sesame
PASSWORD = open sesame
PASSWORD=Open Sesame
```

If you enter them one after another, the third will replace the first, because of the same variable name. The second, however, is treated as a different variable because of the extra space at the end.

The three values after = are treated as being different from one another. The first and third differ in case and the second differs from the others in spacing.

Maneuvering Environment Variables

Although you can enter a variable and its assigned value into the environment from the command line, other actions, such as passing a value from one variable to another, must be done via a batch file. Try the following to demonstrate this point:

1. Set a new prompt:

```
A>prompt $p$g
A:\>_
```

2. Create a batch file:

```
A:\>copy con testprom.bat
set prompt1=%prompt%
echo prompt %prompt1% > newprom.bat
^Z
```

The first line of the batch file will create a new variable named `PROMPT1` and assign to it the value of the variable named `PROMPT`. The notation of `%PROMPT%` means the value of a variable named `PROMPT`. In our case, `%PROMPT%` has the value of `$p$g` because of step 1. After the batch file's first line is executed, `PROMPT1` will hold the value of `$p$g`.

The second line is to save `prompt` and the value of `PROMPT1` to a new batch file named `NEWPROM.BAT`. After this line is executed, that file will contain `prompt $p$g`. You can use the `TYPE` command to verify that.

3. Run the batch file:

```
A:\>testprom
A:\>set prompt1=$p$g
A:\>echo prompt $p$g > newprom.bat
A:\>_
```

Notice that both `%prompt%` and `%prompt1%` are converted to metastrings. If you display the environment, it shows `PROMPT1=$p$g`.

4. Delete the prompt from the environment:

```
A:\>prompt
A>_
```

5. Run the `NEWPROM.BAT` batch file created in step 3:

```
A>newprom
A>prompt $p$g
A:\>_
```

Notice that the system prompt has been changed. The environment now also contains `PROMPT=$p$g`.

6. Make an assignment on the command line:

```
A:\>set prompt1=%prompt%
```

Nothing happens on the screen. If you go to the environment, you see `PROMPT1=%prompt%`, not `PROMPT1=$p$g`.

7. Delete the prompt from the environment:

```
A:\>prompt
A>_
```

8. Change the prompt:

```
A>echo prompt %prompt1%
prompt %prompt1%
A>_
```

The system prompt is not changed; only the literal string is echoed to the screen. Compared to a command entered from a batch file in step 3, the identical command entered on the command line in step 6 does not pass a value from one variable to another.

Selecting a Prompt Style

Parameters (command-line arguments) entered with a batch file (when a batch file is run) can be passed to environment variables, which in turn can pass values to other batch files or be maneuvered in various ways. We will use this mechanism to maneuver prompt styles.

In Chapter 5 you learned to use PROMPT to create a great variety of prompt styles. To create a prompt style, you have to go through laborious steps. Now that you know the environment, you can put a few styles there and use a batch file to make one of them active at a moment's notice.

We will use two batch files. The first is to quickly enter into the environment all the prompt styles you might use. A batch file like this will do:

```
A>copy con envprom.bat
@echo off
set pd=$d$g
set pt=$t$h$h$h$h$h$g
set pp=$p$g
set pv=$v$g
^Z
```

All of these can be individually entered from the command line, if you so wish. The four prompt styles respectively display date, time, current directory, and DOS version. Running this batch file will enter the following variables and values into the environment:

```
PD=$d$g
PT=$t$h$h$h$h$h$h$g
PP=$p$g
PV=$v$g
```

When you want to use a particular prompt style, just run the batch file below with one of the four available parameters:

```
A>copy con setprom.bat
@echo off
if %1.==d. prompt=%pd%
if %1.==t. prompt=%pt%
if %1.==p. prompt=%pp%
if %1.==v. prompt=%pv%
^Z
```

The IF batch command is used to test the parameter entered on the command line. If you enter D as the parameter, for example, the first line is tested true and the assignment is made. Chapter 11 explains this and other batch commands.

If you run this batch file with no parameter or a nonmatching parameter, nothing happens. If you enter a parameter matching one of the four available choices, the matching prompt style is activated. If you go through all four parameters one after another, this is the result:

```
A>setprom t
21:26>setprom d
Sun 06-27-1990>setprom p
A:\>setprom v
MS-DOS Version 4.01>_
```

If you use numerous and multilevel directories, you can use a technique like this to navigate among them quickly. Just enter various directory lists into the environment and use a batch file to quickly make one of them active.

Adding to a Path List

When you enter a variable into the environment, an existing variable of the same name and its assigned value will be erased—just as when you create a new file using

the name of an existing file. What if you just want to add something to an existing variable? You could enlist the help of a batch file.

One occasion when you could use addition rather than replacement is when you want to add a drive or directory to a path list already in the environment. The batch file below could do just that:

```
A>copy con addpath.bat
@echo off
set path=%path%;%1
^Z
```

This batch file will let you keep the old list and add new items to it, instead of completely replacing the old list. Every time you need to add a path, just enter an appropriate parameter after the batch file name, such as:

```
A>addpath b:\ss\acct
```

This will insert a semicolon and the additional path after the existing path list. Suppose the original list is this:

```
PATH=a:\dos\util
```

After running the batch file, use the SET command to show the new list as:

```
PATH=a:\dos\util;b:\ss\acct
```

If no path is specified after the batch file name, such as:

```
A>addpath
```

the batch file will insert an additional semicolon after the original list. This will do no harm.

If you specify the /E switch when you run the APPEND command for the first time, as explained in Chapter 5, this command enters the environment and can thus be manipulated just like PATH. Everything we have said about PATH can be applied to APPEND.

The environment can serve many useful purposes when you write batch files. We will later explore the technique of providing for a password to prevent unauthorized entry to a batch file. See PASS.BAT in Chapter 11 for details.

Practice 9-4

- ☐ 1. Install APPEND with the /E switch. Create a batch file named ADDAPPND.BAT similar to ADDPATH.BAT.
- ☐ 2. Enter an APPEND list into the environment. Use ADDAPPND.BAT to add to the list.
- ☐ 3. Modify ENVPROM.BAT and SETPROM.BAT to display the following prompt:

```
Enter a command:_
```

☐ 4. Create two batch files named `ENVDIR.BAT` and `C.BAT` similar to the above two files and use them to move from one directory to another. Hint: `C.BAT` should use `IF`, and `ENVDIR.BAT` should contain lines like this:

```
set d1=cd\wp\work
```

Drill

_____ 25. The DOS commands that can be entered into the environment include:
- a. `PATH`
- b. `PROMPT`
- c. `APPEND`
- d. all the above

_____ 26. To enable you to enter `APPEND` into the environment, you must run it with the `/E` switch first. True or false?

_____ 27. The command `A>path;` will:
- a. delete the `PATH` string from the environment
- b. enter the `PATH=` string in the environment
- c. do nothing
- d. generate an error message

_____ 28. The command `A>prompt;` will:
- a. delete the `PROMPT` string from the environment
- b. enter the `PROMPT=` string in the environment
- c. do nothing
- d. generate an error message

_____ 29. The command `A>set p1=%prompt%` entered on the command line will:
- a. assign the old prompt value to `P1`
- b. assign `%prompt%` to `P1`
- c. change the prompt on the screen
- d. delete an existing prompt string in the environment

_____ 30. The command `set p1=%prompt%` from a batch file will:
- a. assign the old prompt value to `P1`
- b. assign the `%prompt%` string to `P1`
- c. change the prompt on the screen
- d. delete an existing prompt string in the environment

Summary and Review

Check each item as you review and understand the following highlights of the chapter:

1. Use MEM to find out memory use and availability. Use /PROGRAM or /DEBUG to display detailed information about programs loaded, their sizes and locations.
2. Use MODE without any parameter to report the status of all devices. To report a device's status, put the device name after MODE.
3. Use MODE to set a higher keyboard speed. This will increase your productivity.
4. Use MODE to set monitor modes, including numbers of columns and lines. You need special equipment to set some of the unusual attributes.
5. Use MODE to set a dot-matrix printer to print 80 or 132 columns and 6 or 8 lines per inch.
6. Use MODE to reroute one device to another, such as rerouting printing to a printer connected to a serial port.
7. Use ASSIGN, JOIN, and SUBST to assign one drive to another, join an empty directory to a drive, and substitute a drive for a directory. You can use a logical but nonexisting drive if you add the LASTDRIVE command in your CONFIG.SYS file.
8. Include BREAK, BUFFERS, FILES, DEVICE commands in your CONFIG.SYS file to enhance your system's performance.
9. DOS sets aside a special memory area called the environment to handle the PATH, PROMPT, and APPEND (if installed with /E) commands.
10. The default size for the environment is 160 bytes. If you need more space in the environment, include the SHELL command, plus the /P and /E:bytes switches, in CONFIG.SYS.
11. You can create your own environment variables and assign them values. To maneuver these variables and their values, you can do it with batch files, not on the command line.
12. To take full advantage of the environment, you can enter available directory lists or prompt styles into the environment and then use a batch file to quickly activate one of them. You can also use a batch file to add (not replace) paths to an existing path list.

Review Questions

Write an answer for each question below:

1. What is the major difference between MEM's /PROGRAM and /DEBUG switches?
2. What can MEM supply that CHKDSK cannot?
3. List the devices that MODE can control and their corresponding device names recognized by DOS.
4. What kind of monitor do you have? What functions can you set by using MODE?
5. With a dot-matrix printer, what are the legal numbers of lines and columns you can set by using MODE?
6. What items does a complete device status report contain?
7. Explain the consequence of the following command:

```
A>assign b=a
```

8. What does the LASTDRIVE command do?
9. Explain the differences between JOIN and SUBST.
10. What is the default status of BREAK? What difference does it make whether it is on or off?
11. What does BUFFERS do? Can you enter it on the command line?
12. If you do not specify a value for FILES, how many files can DOS handle at one time? How can you change this value?
13. What is a RAM disk? How can you create one?
14. What is the DOS environment? What is its default size? How can you increase it?
15. What DOS commands are automatically entered into the environment after they are issued? How can you add variables into the environment?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
MEM	Not available	New command
MODE	No keyboard speedup option	New keyboard speedup option
	Can't set number of display lines	Can set number of display lines on EGA and VGA
BUFFERS	No look-ahead or placement of buffers in expanded memory	New look-ahead feature and buffers can be placed in expanded memory
ANSI.SYS	No ability to deal with the enhanced keyboard or extra display lines	New /L switch to increase display lines; /X and /K to enable and disable extra function keys
VDISK (IBM) or RAMDRIVE (MS)	Can create a RAM disk in extended memory	RAM disk can be in extended or expanded memory

Using Edlin

Chapter 10

TOPICS:

A Quick Tour through EDLIN
EDLIN Commands
Inserting Lines
Editing Lines
Moving, Deleting, and Copying Lines
Displaying Lines
Search and Replace
Handling Control Characters
Handling Large and Special Files
EDLIN and Batch Files

COMMANDS COVERED:

EDLIN commands: `A`, `C`, `D`, `E`, `I`, `L`, `M`, `P`, `Q`, `R`, `S`, `T`, `W` (see Table 10.1 for explanation)

NEW TERMS AND SYMBOLS:

line editor, EDLIN (command) prompt, *, current line, ., #, ?, +, -, ^Z, control characters

REQUIRED MATERIALS:

1. Two practice disks
2. Access to EDLIN

Quick Start

✎ **To start EDLIN:**
A>edlin filename

✎ **To end EDLIN and save the edited file:**
Enter E on the command line (after *).

✎ **To quit EDLIN and abandon the edited file:**
Enter Q on the command line and then N at the Y/N prompt.

✎ **For a summary of EDLIN commands:**
See Table 10.1.

✎ **To begin file creation or insert a new line:**
Enter I alone or followed by a line number.

✎ **To show lines without changing the current line:**
Enter L alone or precede it with a range of numbers.

✎ **To show lines and change the current line:**
Enter P alone or precede it with a range of numbers.

✎ **To edit a line:**
Enter the number of that line.

✎ **To speed up editing a line:**
See Table 10.2 for DOS editing keys.

✎ **To delete lines:**
Enter D preceded by a range of numbers.

✎ **To move lines:**
Enter M preceded by a range and destination.

✎ **To copy lines:**
Enter C preceded by a range and destination (and optional number of times).

✎ **To search for a string:**
Enter S preceded by a range and followed by a string. Place ? before S (or R) to show O.K.? prompt.

✎ **To replace a string:**
Enter R preceded by a range and followed by a search and a replace string separated with ^Z.

✎ **To enter a control character:**
See Table 10.3.

EDLIN is the **line editor** program that comes with the DOS package. It can be used to create and edit a text file, and is relatively easy for anyone to master.

Three major categories of software packages are used to maneuver text, namely word processor, text editor, and line editor. A full-feature word processor such as WordPerfect and Microsoft Word provide all imaginable features to manipulate text. A text editor is less sophisticated. The editor in the Microsoft Quick language series is a good example. It has full-screen editing capabilities. But there is no capability to format text output—only plain ASCII text is produced. A line editor like EDLIN also produces only ASCII files, but it can handle only one line at a time, not the whole screen.

EDLIN has a number of pluses and minuses. Let us look at the disadvantages first:

- ✎ **There is no wordwrap. You must press Enter at the end of each line. A line can be no more than 253 characters long.**
- ✎ **You cannot move the cursor up and down. Since EDLIN is a line editor, not a full-screen editor, you can move the cursor or make corrections only in the current line.**
- ✎ **There are limited cut-and-paste capabilities. Although you can easily move and copy entire lines, you need to go through rather elaborate steps to maneuver characters inside a line or across lines.**
- ✎ **There is no undo feature. If you make a mistake, you cannot undo it as you can in other packages.**

Here are the advantages:

- ✎ **EDLIN is easily available. Since it comes with the DOS package, you need not spend extra money to acquire it.**
- ✎ **It is easy to learn. Since there are few complicated features to learn, you can quickly master it.**
- ✎ **It is compact. Its small size will enable you to work with a computer with limited memory and disk space.**
- ✎ **It is convenient for some occasions. Since it outputs text in only the ASCII format, it can be used conveniently to create and edit a DOS batch file or language code. If you have a C compiler without an editor, for example, you can use EDLIN to write C code, which can then be compiled by the compiler.**

EDLIN is particularly useful for creating and editing a long batch file. For a short batch file, you can use `COPY CON`. For a longer one, it is more convenient to use

EDLIN. For batch files containing control characters, you have to use EDLIN because you cannot enter these characters through `COPY CON`.

A Quick Tour through EDLIN

✎ **Start EDLIN by entering it with a file name. Use `I` to start inserting lines and Ctrl-C to end. Enter `E` to save the file and exit. To quit without saving the file, enter `Q` and then Y.**

Using EDLIN requires some simple steps. Let us try a quick tour through it. We will run it, create a simple file, and exit. Follow these steps:

1. Put the DOS disk or another disk containing the `EDLIN.COM` program in drive A.[1]

2. Put a disk in drive B to save a text file.

3. Start the program:

```
A>edlin b:twain
```

If this is a new file or `TWAIN` is not found in drive B, the screen shows the following:

```
A>edlin b:twain
New file
*_
```

The asterisk (*) is the **EDLIN command prompt**, comparable to the > sign for DOS. When it appears at the left margin of the screen, you are to enter one of the available commands to instruct EDLIN to do something.

4. Since we intend to insert text, enter the I command. The screen displays the following:

```
A>edlin b:twain
New file
*i
     1:*_
```

The cursor stays after the * of line 1. The * in a text line denotes the current line (explained below). EDLIN is now ready for you to enter text in line 1.

5. Type the following lines and press Enter at the end of each:

```
1:*East is East, and West is West,
2:*and never the twain shall meet.
3:*_
```

[1] If you want to save your file on the same disk containing EDLIN, you can enter a command like this:

```
A>edlin filename
```

6. Press Ctrl-C (or Ctrl-Break). The cursor moves down and left, and a new * prompt appears. The screen now shows this:

```
A>edlin b:twain
New file
*i
   1:*East is East, and West is West,
   2:*and never the twain shall meet.
   3:*^C

*_
```

7. Enter E to save the file and exit.

The screen shows the A> prompt again. You have now exited EDLIN and created a new file. To verify that, enter the following command:

```
A>type b:twain
```

The screen shows the two lines of text you entered, minus the numbers, asterisks, and indentation.

Note: If anything went wrong when you followed these steps, one of the two things may have happened. First, you may not have entered a required file name. If you do not enter a file name after the program name, EDLIN will not load. You will get this error message instead:

```
File name must be specified
```

Second, you may not have provided the path for DOS to find the program. This situation will lead to this message:

```
Bad command or file name
```

If this happens, you must direct DOS to the drive or directory storing EDLIN.COM.

Hard Disk Users: If you have a hard disk with EDLIN stored in the DOS directory, you can issue this command no matter what the default drive or current directory is:

```
c:\dos\edlin a:twain
```

With this arrangement, EDLIN will be loaded from the specified directory and your file will be saved to the disk in drive A. If you do not specify a directory for your text file, it will automatically be saved to the current directory of the default drive, not necessarily where EDLIN resides.

If you do not want to save a file, enter the Q command instead of E. You will be asked this:

```
Abort edit (Y/N)?_
```

If you press Y, the DOS prompt appears and the file is not saved. If you press any other key, you are back to EDLIN with the file still in memory.

EDLIN will automatically insert a ^Z marker at the end of any file it saves. This is in contrast to COPY CON, which by default does not add such a marker to the end

of a file it creates. EDLIN can edit a file created with `COPY CON`. However, a `^Z` will be added to the end, even though it was not present in the original.

EDLIN Commands

✎ **This section shows a table summarizing all the EDLIN commands. Use it for quick reference.**

Now that you know the rudimentary steps of using EDLIN, let us explore its other features. Table 10.1 shows what command to enter to accomplish a specific purpose.

A glance at Table 10.1 should convey the impression that an EDLIN command consists of one letter (in uppercase or lowercase), plus optional parameters (numbers and signs) placed mostly before but sometimes after the letter.

As EDLIN commands are cryptic, you may need to refer to this table frequently to enter correct commands. Each command's details will be explained more thoroughly in this chapter.

Inserting Lines

✎ **Enter `I`, preceded by a line number, to insert a line at that location. Enter `I` alone to insert at the current line (marked with *).**

✎ **Enter `L` to list (display) lines.**

Let us now retrieve the file we just created and make some additions:

1. Run EDLIN to edit an existing file:

```
A>edlin b:twain
End of input file
*_
```

The message tells you that the entire file has been retrieved to the memory, and the * prompt is ready for your input.

2. Enter the L (List) command to display the file:

```
A>edlin b:twain
End of input file
*l
       1:*East is East, and West is West,
       2: and never the twain shall meet.
*_
```

Table 10.1 EDLIN Commands

Command	Format	Purpose
A	nA	Append from disk n lines to the file in memory being edited
C	m,n,p,q	Copy m through n lines to p*th* line q times; p*th* becomes current
D	D	Delete current line
	nD	Delete n*th* line
	m,nD	Delete lines m through n
	,nD	Delete from current to n*th* lines
E	E	Exit with file saving
I	I	Insert at current line
	nI	Insert at n*th* line
	#I	Insert at end of file
L	L	List current plus 11 above & 11/12 below
	nL	List 23/24 lines from n*th* line
	m,nL	List lines m through n
	,nL	List current to n*th* line
M	m,n,qM	Move lines m through n to q*th*
P	P	Page from below current line
	m,nP	Page m through n
	,nP	Page from below current through n
Q	Q	Quit without saving file
R	m,n?Rstring1 ^Zstring2	Search m to n lines and replace `string 1` with `string 2`; optional ? displays `O.K.?`
S	m,n?Sstring	Search m to n lines for `string`; ? displays `O.K.?` prompt
T	nTdrive:filename	Transfer (merge) file on disk to memory, enter at n*th* line
W	nW	Write n lines in memory to disk
n	n	Enter number n to edit n*th* line
Enter	Enter	Press Enter alone to edit one after current line
(The following are used with other commands)		
.	.	Denotes current line
#	#	Denotes end of file, after last line
?	?S	Used with `S` & `R` to display a prompt
-/+	-5,+5L	Number of lines before/after current one
^Z	Rs1^Zs2	Separate search and replace strings

(Note: m, n, p, and q denote line numbers.)

Notice that the first line is preceded with an *. It tells you that this is the current line. The **current line** is always line 1 when a file is first retrieved. The current line changes as different commands are issued. Many commands will be applied to the current line by default.

3. Enter the `I` command to insert a new line at the current line, namely the first line; you can also use `1I` for inserting at line 1.

4. Type `Rudyard Kipling` and press Enter three times, the first to end the line, the second to create an empty line, and the third to move the cursor to a new line.

5. Press Ctrl-C to end editing. The screen should look like the following:

```
A>edlin b:twain
End of input file
*l
       1:*East is East, and West is West,
       2: and never the twain shall meet.
*i
       1:*Rudyard Kipling
       2:*
       3:*^C

*_
```

6. Use the `L` command to list the lines:

```
*l
       1: Rudyard Kipling
       2:
       3:*East is East, and West is West,
       4: and never the twain shall meet.
*_
```

Notice that the original lines have been moved down and renumbered. Also, the third line has become the current line (marked with *), due to the fact that the previous editing ended at that point. If you now issue the `I` command alone, this particular line will appear.

7. Issue the `#I` command to insert at the end:

```
*l
       1: Rudyard Kipling
       2:
       3:*East is East, and West is West,
       4: and never the twain shall meet.
*#i
       5:*_
```

8. Add the additional lines as shown below:

```
1: Rudyard Kipling
2:
3: East is East, and West is West,
4: and never the twain shall meet.
5:
6: Bible
```

```
 7:
 8: Do unto others what you want
 9: others to do unto you.
10:
11: Confucius
12:
13: Do not do unto others what you
14: do not want others to do unto you.
15:
16: Kipling, the old British imperialist, was
17: right. The two civilizations will travel
18: different paths. Just look at the popular
19: quotes from the Bible and the Chinese sage
20: Confucius. The former implores people to go out
21: and convert the infidel, and the latter
22: counsels people to leave others alone.
23: Even the simplest idea of being kind to
24: your fellow men can take such different
25: paths.
```

If you break (pressing Ctrl-C) at line 26 and then use the L command to display the lines, the * showing the current line will not be seen anywhere. The reason is that the line below the last line is designated as the current line. Since that line does not exist, it is not displayed, nor is the *. Details of displaying lines will be discussed later.

9. Enter the E command to save the file and exit.

The new TWAIN file now contains the above 25 lines of text. What happened to the old file? It has been renamed TWAIN.BAK. The contents are unchanged; only the name is altered. If the original file name has an extension, such as TWAIN.DOC, the extension will be replaced with BAK. EDLIN always does this whenever you save an existing file.

Tips & Tricks

Editing a Backup File

Should something happen to a text file and you can no longer use it, you can use the BAK file. However, you cannot retrieve it with this name. If you try, you will get this error message:

```
Cannot edit .BAK file—rename file
```

If you change the name to anything that does not have the BAK extension, you can retrieve it to EDLIN. By accessing the backup file this way, you may salvage some data, but perhaps not all.

Version 3.3, unlike others, does allow you to edit a BAK file. It is, however, a good idea to change to another name before editing.

Drill

_____ 1. You can start EDLIN by entering it alone, like this: `A>edlin`. True or false?

_____ 2. In most versions of DOS, you can retrieve to EDLIN a file with a `BAK` extension. True or false?

_____ 3. All files created with EDLIN end with a `^Z` marker. True or false?

_____ 4. The current line is marked with * at the beginning. True or false?

Questions 5–8 are all related to the `I` command.

_____ 5. To insert a line at the current line, just enter `I`. True or false?

_____ 6. To insert at the end of a file, enter `#I`. True or false?

_____ 7. If you insert at line 5, the original text in line 5 is deleted and replaced by the new entry. True or false?

_____ 8. To stop inserting lines, press Ctrl-Z. True or false?

_____ 9. If you find the message `End of input file` at the beginning of the EDLIN screen, it means that:
a. this is a new file
b. part of an existing file is retrieved
c. this session is terminated
d. the entire file is retrieved

Editing Lines

- **Edit a line by entering that line's number. Enter a period (.) to edit the current line. Press Enter alone to edit the line next to the current line.**
- **To designate a new current line, enter its number and press Enter to end editing.**
- **Use DOS editing keys to speed up editing a line.**

When you want to edit a line, you need to enter that particular line's number. The line is displayed and a blank line appears below for you to edit. Suppose you want to edit line 16. You can enter that number to get the following:

```
*16
        16:*Kipling, the old British imperialist, was
        16:*_
```

You can now do a number of things. If you press Enter without typing any text, the original line is kept and editing is terminated. This line (16 in our example) has also become the current line. This is what you can do if you want to designate a particular line as the current line. To designate the end of the file, the line below the last existing line, as the current line, use # in lieu of a number.

You can use DOS editing keys to speed up your work. Chapter 4 explained how these keys work. Table 10.2 shows these keys and their purposes.

If something you want to change lies near the end of a line, you can edit it this way: Enter the line number, press F3 to display the whole original line, press Backspace or the Left Arrow key a number of times, type the new characters, and press Enter.

If errors lie somewhere else, you may try some other ways. These may seem a little complicated at first. But if you can master them, it is less time-consuming than retyping the whole or most of a long line.

Table 10.2 DOS Editing Keys

Key	Function
F1 (or Right Arrow)	Copy one character from template to screen
F2c	Copy all characters before first typed c
F3	Copy all characters from cursor to end of line
F4c	Skip characters before first typed c (then use F1 or F3 to copy the rest)
F5	Store the current line in the template
Esc	Abandon edited line; start anew
Ins	Insert mode on/off
Del	Skip over a character in template
Backspace (or Left Arrow)	Delete to left a displayed character

(c denotes a character)

As an example, let us replace "old" with "noted" in line 16. These are the simplest steps using the DOS editing keys:

1. Enter 16 to edit it. When the line appears, press F2 and then o. The part before the character is displayed, and the cursor stays at the location of the typed character:

```
16:*Kipling, the _
```

2. Type `not` in the location of `old`. Since `noted` is longer than `old`, you need to turn on the Insert mode. So press Ins and type `ed` to complete the word.

3. Press F3 to copy the rest of the line. Press Enter to complete editing. The screen shows this result:

```
*16
        16:*Kipling, the old British imperialist, was
        16:*Kipling, the noted British imperialist, was
*_
```

If you did not press Ins in step 3, the outcome would be:

```
16:*Kipling, the notedritish imperialist, was
```

This line has the exact amount of space as the unedited line, just different characters, some of which have replaced those in the template.

What can you do if you replace a longer word with a shorter one? Instead of turning on the Insert mode, you use Del to skip over the extra characters of the longer word.

Suppose you want to replace `noted` with `old` in line 16. First, you need to move quickly to the beginning of the word to be replaced. To do that you can press F2, followed by n, and then repeat another time to reach the second occurrence of the character n. Or you can press F1 repeatedly until you reach the desired location.

At step 2, type `old` and then press Del twice to skip over the two extra characters in `noted`. If you then press F3 to copy the rest, the edited line becomes:

```
16:*Kipling, the old British imperialist, was
```

If you had not pressed Del twice, the outcome would have been the following instead:

```
16:*Kipling, the olded British imperialist, was
```

The extra `ed` is the remnant of `noted`, which is copied if you do not press Del to skip over it.

Suppose you have a line like the one below. How can you quickly fix it?

```
16:*Kipling, the notedritish imperialist, was
```

You can follow these steps:

1. Enter 16 to edit the line.
2. Press F2, followed by r, to display the following:

   ```
   16:*Kipling, the noted_
   ```

3. Press Ins, the space bar, and type B.
4. Press F3 to display the rest. The result is as follows:

   ```
   16:*Kipling, the noted British imperialist, was
   ```

   If you had not pressed Ins in step 3, the result would have been:

   ```
   16:*Kipling, the noted Btish imperialist, was
   ```

Here the line is not expanded, and the new characters have replaced the old ones.

If you decide to abandon editing a line and want to start the line at the beginning, just press Esc. A backslash (\) appears at the end of the line. This line is not registered in the template and has no effect on the file. The cursor goes to the beginning of the next line for you to enter text, as shown below:

```
16:*Kipling, the noted British imperialist, was
16: Kipling, the noted\
    _
```

If you type text and press Enter, this line replaces the old one. If you type no text and just press Enter, the old line (unedited) is kept.

EDLIN makes it relatively simple for you to edit the current line and the line below it. To edit the current line, simply enter a period (.), instead of a line number, at the command prompt. To edit the line after the current one, just press Enter at the * prompt. Using this technique, you can quickly edit one line after another. If the current line is after the last existing line, then nothing happens because the line does not exist.

We have explored all the keys in Table 10.2 except F4 and F5. These are not commonly used. Nevertheless, you may find some uses for them.

F4 lets you skip up to a specified character and copy the rest of a line. Suppose you want to get rid of the Kipling name in line 16. These are the steps to follow:

1. Enter 16 to edit the line.
2. Press F4 and t to skip up to the first `t`.
3. Press F3 to retrieve the rest of the line. The result is as follows:

```
*16
          16:*Kipling, the old British imperialist, was
          16:*the old British imperialist, was
*_
```

F5 lets you put a new line in the template without replacing the old line in the file. This affects only the template, not your text line. The following steps illustrate its use:

1. Enter line 6 to edit it. Press F3 to display `Bible` and prove that the template contains this word. Press Backspace to erase the whole word.
2. Type `booboo` and press F5. An `@` sign appears at the end and the cursor moves to the beginning of the line below.
3. Press F3. The `booboo` string is retrieved, proving that it is stored in the template, replacing `Bible`.
4. Press Esc to abort editing of the line. A \ sign appears.
5. Enter 6 to edit the line. The original line appears, proving that the text line has not been altered.

6. Press Enter to return to the command line. The results are as follows:

```
*6
        6:*Bible
        6:*booboo@
           booboo\

*6
        6:*Bible
        6:*
*_
```

Tips & Tricks

EDLIN's Two Buffers

EDLIN maintains two separate buffers, one for the text line (as explained in the text) and the other for the command line.

On the command line, when the cursor is after the * at the left edge of the screen, you can also use the DOS editing keys to maneuver characters previously entered on the command line.

Most of the time you do not need to resort to this because most EDLIN commands consist of one character. However, when a long string is involved, such as a replace operation explained in the next section, this knowledge comes in handy.

Practice 10-1

☐ 1. Use EDLIN to create a file named SAYING, as shown below. Use proper editing keys to avoid typing repetitive words of the second and third lines in each quote.

```
1:  Francis Bacon
2:
3: Reading makes a full man,
4: conference a ready man,
5: and writing an exact man.
6:
7: Abraham Lincoln
8:
9: You can fool some of the people all of the time,
10: all of the people some of the time,
11: but you cannot fool all of the people all of the time.
```

☐ 2. Save SAYING and exit EDLIN.

Drill

_____ 10. If you just press Enter at the * command prompt, EDLIN expects you to:
a. insert at the current line
b. edit the current line
c. edit the line after the current line
d. exit and save the file

_____ 11. If you enter # alone at the command prompt, which of the following becomes the new current line:
a. line 1
b. the line after the present current line
c. the last line
d. the line after the last line

Choose one of these DOS editing keys to answer questions 12–17:
a. F1, b. F2, c. F3, d. F4, e. F5, f. F6, g. Esc.

_____ 12. To copy all the remaining characters from the template.

_____ 13. To copy one character from the template.

_____ 14. To abandon an edited line and start anew.

_____ 15. To skip before a typed character.

_____ 16. To copy before a typed character.

_____ 17. To put a line in the template without affecting the existing line.

Moving, Deleting, and Copying Lines

- **Use M (C), preceded by a range and a target, to move (copy) the range of lines to the target.**
- **Use D, preceded by a range, to delete lines. Enter D alone to delete the current line.**

Moving, deleting, and copying lines are simpler processes than changing words inside a line. Being a line-oriented editor, EDLIN makes it relatively easy to maneuver lines, whether singly or in a block.

To demonstrate the steps of using the M (Move), C (Copy), and D (Delete) commands, let us retrieve the TWAIN file and follow these steps:

1. Display the first 15 lines:

```
End of input file
*1,151
           1: *Rudyard Kipling
           2:
           3: East is East, and West is West,
           4: and never the twain shall meet.
           5:
           6: Bible
           7:
           8: Do unto others what you want
           9: others to do unto you.
          10:
          11: Confucius
          12:
          13: Do not do unto others what you
          14: do not want others to do unto you.
          15:
*_
```

2. Move lines 5–9 to line 15, and display the new lines:

```
*5,9,15m
*1,151
           1: Rudyard Kipling
           2:
           3: East is East, and West is West,
           4: and never the twain shall meet.
           5:
           6: Confucius
           7:
           8: Do not do unto others what you
           9: do not want others to do unto you.
          10:*
          11: Bible
          12:
          13: Do unto others what you want
          14: others to do unto you.
          15:
*
_
```

The 5,9,15M command tells EDLIN to move lines 5–9 to the location of line 15. The Confucius quote moves up to fill in the vacated lines, occupying lines 5–9. Notice that the new current line is the beginning of the moved block, namely line 10.

3. Delete line 16 through the end, and redisplay file:

```
*16,#d
*1l
```

The pound sign (#), you will recall, designates the end of the file. The bottom half of the file has disappeared. The new file contains only the lines shown above, minus the current line marker. The current line is after the last line and is thus not shown.

If you enter `L` alone (instead of `1L`) at this point, you will display only lines 5–15, not all the 15 lines. The reason for this will be explained in the next section.

4. Delete line 1 and redisplay the remaining lines:

```
*1d
*1
```

Every line is renumbered. There are now 14 lines left. The current line is line 1, after the previously deleted line. If you enter `D` alone at this time, line 1 (the current line) will be deleted.

5. Delete the current line through line 9, and display the remaining lines:

```
*,9d
*1
        1: *Bible
        2:
        3: Do unto others what you want
        4: others to do unto you.
        5:
*_
```

If you do not specify the number before the comma, EDLIN assumes that you want to begin at the current line. If the current line is not line 1 and you intend to delete lines 1–9, you must enter the following:

```
*1,9d
```

You can also mark so many lines before and after the current line by using - and +. If you enter this command:

```
*-1,+2d
```

you instruct EDLIN to delete four lines—the current line, one line above (-1), and two lines below (+2).

Warning: One important precaution to keep in mind when deleting lines is that the line numbers below are immediately renumbered. You should then use the `L` command to show the new line numbers before deleting more lines. If you do not, you may end up deleting the wrong lines.

6. Copy the current line (1) to the 7th line three times and display the new lines:

```
*,,7,3c
*l
            1: Bible
            2:
            3: Do unto others what you want
            4: others to do unto you.
            5:
            6: Bible
            7:*Bible
            8: Bible
*_
```

Note: Notice that EDLIN starts filling in the first line after the last existing line, namely line 6. Line 7 is designated the current line, as you have ordered. What happens if you want to copy to line 7 only once? Only line 6 will be filled. Line 7 is still the designated current line, but it is not shown because it does not exist.

You can enter up to four numbers before the C command, each separated by a comma. The first two are for the beginning and ending line numbers. If no number is entered before and after the first comma, the current line is assumed. If there is only one line to be copied, you need to enter the same number, such as:

```
*5,5,10,1c
```

These numbers tell EDLIN to copy line 5 alone to line 10 once. The third number (the destination line) must be entered; otherwise, an error message will be displayed. If you do not specify the last number, EDLIN assumes that you want to copy only once. If line 5 is the current line, the above command could be simplified to this:

```
*,,10c
```

7. Finally, terminate editing without saving the file:

```
*q
Abort edit (Y/N)?y
A>_
```

The earlier file, which you retrieved at the beginning of this session, remains intact. The mutilated file on the screen is abandoned. If you use the E command at the end, this mutilated file will be saved, and the original file will be renamed with a BAK extension.

Warning: A final word of caution. Before you delete lines, make sure that is what you intend to do. After you press Enter to register your command, the lines are gone forever. You can no longer get them back, unless you go back to the old file still saved on disk. Some word processors, such as WordPerfect, allow you to undelete (retrieve from the buffer) up to three previous deletions. EDLIN, however, lacks that capability.

Drill

_____ 18. If the current line is neither the first nor the last of the file, the `-1,+1D` command will delete:
a. 0 line
b. 1 line
c. 2 lines
d. 3 lines

_____ 19. Entering the `D` command alone will delete only the current line. True or false?

_____ 20. The `1,4,10,3C` command will:
a. copy lines 1–4 to line 10 three times
b. move the above
c. copy lines 4–10 to line 3 once
d. cause an error message to be displayed

_____ 21. The `,,5M` command will:
a. move one line
b. move five lines
c. remove one line
d. remove five lines

Displaying Lines

- **Use `L` to display lines around the current line without changing the current line. Precede it with a range to display those lines.**
- **Use `P` to display lines and change the current line.**

So far you have learned to use only the `L` (List) command to display text lines on the screen. There is also the `P` (Page) command. They behave differently. This section explains how to use both of them.

Using the `L` Command

The `L` command alone is used to display lines around the current line. Under normal circumstances, it will display the current line in the middle and 11 lines above and 11 lines (12 lines for version 4) below. The top portion can be no more than 11 lines, while the bottom part may be more.

When a file is newly retrieved, with the first line designated as the current line, issuing the `L` command will display the current line, plus 22 (23 for version 4) lines below, thus filling the screen with 23 (24 for version 4) lines. The number of lines below the current line depends on the lines above it. If the above lines are fewer than 11, the bottom part is proportionally increased.

If you use L alone, EDLIN will not display more than 11 lines above the current line. For example, if there are 15 lines and line 15 is the current line, using the L command alone will display only lines 4–15.

You can designate a specific range of lines around the current line to display. For example, if you issue this command:

```
*-5,+51
```

you tell EDLIN to display five lines above and five lines below the current line.

If you want to display from the current line, use a period (.). For example, this command

```
*.1
```

will display the current line at the top, followed by 22 (23 for version 4) lines below.

You can use numbers to display a specific range of lines. For example, this command

```
*5,101
```

will display lines 5–10 regardless of where the current line is. The second number must be higher than the first. Otherwise, an error message will be displayed.

You can omit either of the two numbers. If you omit the first, such as:

```
*,151
```

EDLIN assumes that you want to display from 11 lines before the current line and end with the specified line.

If the second number is omitted, the display starts with the first number and ends with 23 (24 for version 4) lines below.

You can combine editing a line with displaying a range of lines. For example, this command

```
*15;-5,+51
```

will let you edit line 15 and, when you press Enter after finishing it, display the edited line, plus five lines above and five lines below.

Notice the semicolon (;) separating the two separate commands. If nothing is entered before the semicolon, the line after the current line is made available for you to edit. If you want to edit the current line, enter a period (.) before the semicolon.

Version Variances: What happens if you enter a large number of lines to display on the screen? Suppose you enter this:

```
*1,501
```

Version 3.3 and before will scroll through the whole thing. The earlier lines will scroll out of view and end with the last 23 lines on the screen. If you have a large file and want to scroll from the beginning to the end, use this command:

```
*1,#1
```

If you want to pause the scrolling at any point, press the Pause key or Ctrl-S (or Ctrl-Num Lock); press any key to continue the scrolling. You can also press Ctrl-C or Ctrl-Break to break the scrolling.

Version 4 has added a new twist. The screen will display 24 (rather than 23) lines at one time. If there are more lines to be displayed, the scrolling will pause and this message will be displayed at the bottom:

```
Continue (Y/N)?_
```

If you press N, the scrolling is terminated. If you press Y, another screenful of lines are displayed, forcing the earlier lines to scroll out of sight.

Using the P Command

All that is said so far about the L command can be applied to the P command, with two major exceptions. First, P behaves slightly different from L. Second, P changes the current line.

Normally, the P command alone displays 23 lines starting after the current line, except when the current line is 1. When line 1 is the current line, entering P will display lines 1–23. If the current line is 5, however, P will display lines 6–28, or the end of file if there are fewer lines.

Every time P is used, the current line is changed to the last displayed line. Thus, if the current line is 1 and you continue entering P alone, you can page through the whole file 23 lines at a time.

Version Variances: When a range is specified, P behaves exactly like L, displaying only a specified range of lines. For versions before 4.0, scrolling continues until the end. For version 4, 24 lines are displayed and a `Continue (Y/N)?` prompt appears. In all versions, the last line of the range becomes the current line.

Practice 10-2

- ☐ 1. Retrieve SAYING to EDLIN.
- ☐ 2. List all the lines.
- ☐ 3. Move the first quote below the second.
- ☐ 4. Use P to display the lines.
- ☐ 5. Exit without saving.

Drill

_____ 22. If you enter the L command alone, you
  a. always list from line 1
  b. list lines above and below the current line
  c. display the next screenful of lines
  d. always display 23/24 lines

_____ 23. Using the P command will alter the current line. True or false?

_____ 24. You can repeatedly use the P command alone to page through an entire file. True or false?

_____ 25. If line 1 is not the current line, entering the P command alone will cause EDLIN to display 23 lines of text starting from and including the current line. True or false?

Search and Replace

- **Use S, preceded by an optional range and followed by a search string, to search for matching lines.**
- **Use R, preceded by an optional range and followed by a search and a replace string separated by ^Z, to search for the former and replace it with the latter.**
- **Place a ? before S or R to display the O.K.? prompt.**

EDLIN allows you to search for or replace a string. These features are easy to use and quite speedy.

Searching for Matching Lines

Let us demonstrate how to search for a string in our file. Retrieve the TWAIN file and use the L command to show the first part. Then follow the steps shown below:

```
*snot
       13: Do not do unto others what you
*s
       14: do not want others to do unto you.
*s
Not found
*_
```

The first command, snot, tells EDLIN to search for the string of not. After that command is entered, line 13 appears. This line becomes the current line.

The second command, S alone, continues to search from below the current line for the same string last entered. This search results in the display of line 14.

The third command results in displaying the Not found message. The current line is not changed from the previous search.

Now enter 1 to designate line 1 as the current line and try another way to search for the same string. Enter other commands as shown below:

```
*1
       1:*Rudyard Kipling
       1:*
*?s
```

```
        13: Do not do unto others what you
O.K.? n
        14: do not want others to do unto you.
O.K.? n
Not found
*_
```

The ?, placed before the S command, tells EDLIN to pause and display the O.K.? prompt for user input. If this is the line you are looking for, you are to press Y or Enter. The search then stops. If you press anything else, including the space bar, the search continues.

If you want to search in a specific range, place the line numbers before the S command, such as:

```
*5,20snot
```

If the first number is omitted, the default is the line below the current line. If the second number is missing, the last line is assumed. If you want to search the entire file with the prompt, enter 1 as the only number before S, such as

```
*1?snot
```

EDLIN's search is case sensitive. EDLIN will only identify strings that literally match the search string you enter. If anything does not match, case or spelling, EDLIN cannot find what you want. Thus the following search strings are not the same:

```
*1?sbible
*1?sBible
```

A search string can be anything, including spaces and control characters as explained below. For example, you can have the following as a search string:

```
*1?sBritish imperialist
```

The string between S and the place where Enter is pressed is what EDLIN will search for in your file.

Replacing Strings

Replacing a string follows the same rules as a search operation, with the exceptions of substituting R for S and adding a replace string. The search string is placed ahead of ^Z (Ctrl-Z or F6) and the replace string after it.

To demonstrate replace operations, enter the following two commands:

```
*16,16rimperialist^Zcolonialist
        16:*Kipling, the noted British colonialist, was
*16,16rnoted ^Z
        16:*Kipling, the British colonialist, was
*_
```

Both commands limit operations to line 16. The first replaces imperialist with colonialist, and the second replaces noted with no string, thus erasing it. Notice that the noted string has an extra space. Without it, an extra space will be left after the deleted word.

A replace operation may seem complicated. It is actually very simple. First you specify a range of numbers, which is optional (if no number is specified, the replace range is from below the current line to the end of the file). Then enter the R command,

then the search string, then the ^Z character. Finally, enter the optional replace string (if no replace string is specified, the search string, when found, is deleted).

You have the option to insert spaces before R for easier reading. After R, however, space and no space are treated differently. The following are the same:

```
*1 rBible^Zbible
*1RBible^Zbible
```

These two, however, are not the same:

```
*1rBible^Zbible
*1rBible^Z bible
```

The second command inserts an extra space in the original string, besides converting the first letter to lowercase.

Like a search string, a search-replace string is saved in the buffer. If you use R alone after a replace operation, EDLIN will repeat the previous operation. This may produce unexpected results, possibly mutilating your file beyond recognition.

After a replace operation, you can also use S alone to search for the search string entered in the replace operation. For example, if you did this:

```
*6,6rBible^Zbible
```

you can use S alone to search for Bible.

If you need to alter a long search or replace command, try one of the DOS editing keys explained earlier. This can save you time in entering a long command.

Warning: One last word of caution. Doing a global replace operation, from line 1 to the end and without a prompt, is dangerous. Your file could be ruined. Fixing such a file will be a heck of a headache. It is thus advisable to use a prompt (?) when doing a replace operation in a large range of lines, such as the following:

```
*1?rBible^Zbible
        6: bible
O.K.? n
       19: quotes from the bible and the Chinese sage
O.K.? y
*_
```

EDLIN will show you what the altered line will look like and let you choose yes or no. If you press Y or Enter, the displayed line will be registered. If you press any other key, the original line will be kept.

Practice 10-3

- ☐ 1. Load EDLIN and retrieve SAYING. Use R to delete both first names.
- ☐ 2. Delete the entire file.
- ☐ 3. Exit and save the file.
- ☐ 4. Delete SAYING and rename SAYING.BAK as SAYING.
- ☐ 5. Reload EDLIN and retrieve SAYING.

☐ 6. Quit EDLIN without saving the file.

Handling Control Characters

Table 10.3 Control Characters

ASCII	Display	Purpose	Key(s) Pressed
0	^@	Null character	F7
7	^G	Sound bell	Ctrl-G
8	^H	Backspace	Ctrl-V H
9	^I	Horizontal tab	Ctrl-V I
10	^J	Linefeed	Ctrl-V J
11	^K	Vertical tab	Ctrl-V K
12	^L	Formfeed	Ctrl-V L
13	^M	Carriage return	Ctrl-V M
27	^[	Escape character	Ctrl-V [

✎ **Insert a control character in an EDLIN file by placing the character after ^V.**

Control characters, those special ASCII codes numbered 0 to 31, can be entered in a batch file to control different parts of a computer system, particularly the printer. What these characters do will be further explained in Chapters 11 and 12. This section mostly illustrates how you can enter them in EDLIN.

The control characters can be entered only in EDLIN (or other software packages), but not COPY CON. This in turn makes it valuable in creating useful batch files. Table 10.3 shows the most commonly used control characters.

Entering most of these characters requires special steps. For example, ^@ cannot be entered in any other way except by pressing the F7 key. Others can be entered by holding down Ctrl and pressing an appropriate letter, such as Ctrl-G.

Characters that cannot be entered by combining Ctrl with a letter can be entered by inserting Ctrl-V before them. Follow these steps to enter ^H:

1. Use the I command to begin inserting in line 1.
2. Hold down Ctrl, press v, and release Ctrl to get ^V on the screen. Hold down Shift and press h to get H. You must use uppercase here.
3. End insertion by pressing Ctrl-C.
4. Use the L command to display H^ on a Hercules monographics screen (^H on a VGA monitor).

5. Edit line 1 and press F1. The screen shows ^H. That is the ^H (Ctrl-H) character DOS will recognize. The screen (Hercules) shows the following after all these steps:

```
*i
       1:*^VH
       2:*^C

*1
       1: H^
*1
       1:*H^
       1:*^H
*_
```

Some of the control characters are used to control the keyboard. That is why you need to use ^V to enter them into a file. If you press Ctrl-H, for example, the cursor will erase to the left, behaving like the Backspace key. To tell EDLIN to put these keystrokes in the text file, you need to enter ^V and then H.

The same is true for ^C. If you press ^C in a line, you terminate editing immediately. What if you want to put the character in the text without terminating editing? You put a C after ^V.

In ANSI.SYS, as explained in Chapter 12, you can use $e[as the escape character. This character is placed before certain instructions to the computer to perform a specific act. You can enter this in EDLIN or use the alternative of ^[[.

Tips & Tricks

EDLIN, ANSI.SYS, and Control Characters

As explained in Chapter 12, the Esc [7m command instructs your monitor to display text in reverse video. To enter the escape character (Esc), you need to press Ctrl-V, followed by typing [[. The ^V and [combination is treated as ^[by EDLIN and as the escape character (Esc) by DOS. The next [is required by ANSI. After these preliminaries, DOS can interpret the 7m as ordering reverse video. The steps below for creating this file are similar to those enumerated before:

```
*i
       1:*prompt ^V[[7m
       2:*cls
       3:*prompt
       4:*^C

*e
```

After creating this batch file, you can run it to display a color or monochrome monitor in reverse video. If your CONFIG.SYS file does not include ANSI.SYS, the first line will be treated as a meaningless text string, which will be displayed and cleared (by line 2) without changing the monitor display.

Handling Large and Special Files

✎ **Use `W` (Write) and `A` (Append) to handle large files. Use `T` (Transfer) to merge a disk file with the one in the memory.**

EDLIN is suitable for handling small and medium files, particularly batch files that are rarely large. If you must handle a large file with EDLIN, you need to be aware of some commands to use in case you get in a jam.

Working with Large Files

With a large file and/or limited memory, EDLIN retrieves only up to 75% of the available memory. If you do not see the `End of input file` message, that means the entire file is not retrieved.

After editing a file that is not completely retrieved, use the `W` command to save earlier lines, thus freeing some memory. You can specify a number of lines to be written to disk by putting a number before `W`, or let EDLIN free up to 25% of memory by using `W` alone.

After memory is made available, use the `A` command to append more text lines from disk to memory for editing. You can specify a number before `A` to append that many lines, or let EDLIN append to fill up to 75% of memory.

Another important precaution while working with a large file is to make sure you have enough disk space to save the edited file. If there is not enough space when you use the `E` command to exit EDLIN, only a portion will be saved and the saved file named with the `$$$` extension; the rest is lost. The original file is not changed in name or contents.

You can also use the `T` command to transfer or merge a file on disk to the file being edited in the memory. The entry point of the new text can be specified in a line number before `T`, or at the current line if no number is specified.

Editing Binary Files

Some versions of DOS allow you to use the `/B` switch to edit a binary file, such as:

```
A>edlin filename.ext /b
```

Without the switch, EDLIN will operate in ASCII mode and retrieve only the part of the file before the first `^Z`; the rest is ignored.

Other versions, because of bugs, will ignore the `/B` switch and refuse to retrieve after the first `^Z` encountered. One way to remedy this is to retrieve the file, delete it, and merge the original file. After the file you want to edit is retrieved, use the `L` command to show it. If it does not show the entire file, then enter these commands:

```
*1,#d
*tb:filename.ext
```

The first command deletes everything in the memory. The second retrieves the whole file from drive B, including all the `^Z` characters. This works with all versions of PC-DOS and MS-DOS.

> **Warning:** Most of the time you should avoid using EDLIN to edit a non-ASCII text file. If you retrieve a compiled program, such as `MORE.COM`, the screen shows mostly strange characters. If you retrieve a regular WordPerfect file, you find some ASCII characters plus many incomprehensible codes. If you try to edit such a file, you risk the possibility of locking up your computer.

EDLIN and Batch Files

✎ **Use a batch file to speed up using EDLIN to edit another batch file, and to run the edited batch file upon exiting EDLIN.**

If you use EDLIN often, you can use a couple of batch files to speed up your work. This arrangement is particularly useful if you use EDLIN to experiment with a series of batch files.

Suppose you have a batch file named `X.BAT`, which you need to repeatedly edit and run. You could use a batch file named `E.BAT` with which you can quickly load EDLIN and retrieve `X.BAT` for you to edit. Then, when you exit EDLIN, the original batch file will automatically run the edited batch file. All you need to do is to create a batch file like this:

```
A>copy con e.bat
edlin x.bat
x
^Z
```

Assuming that EDLIN is stored on the disk in drive A, running `E.BAT` will automatically load EDLIN and retrieve (or create) the `X.BAT` file for you to edit. When you exit EDLIN, the edited `X.BAT` file will automatically be executed.

If you want to keep the contents of `X.BAT`, you can rename it or copy it to another disk. Otherwise, the edited version will replace the old one on the same disk. The `X.BAT` file will continuously change as you use it to experiment with new batch files.

When you are ready to experiment with another batch file, run `E.BAT`, delete the contents of `X.BAT` (`1,#D` will delete everything), enter new batch lines, and exit to automatically run `X.BAT`. This arrangement can save you a lot of time as you work with lots of batch files in the next two chapters.

Practice 10-4

- ☐ 1. Use EDLIN to create a database of 5 names (last name first) and telephone numbers.
- ☐ 2. Create a similar database with different names. Merge the two databases into one and save it. (Save this file for the next chapter.)
- ☐ 3. Use the `SORT.EXE` program to sort the file.

- ☐ 4. Retrieve the sorted file to EDLIN and examine it.
- ☐ 5. Use EDLIN to write a report to explain how each of the above is done.
- ☐ 6. Create a batch file that will let you quickly access the database, such as looking up a friend's phone number.

Drill

_____ 26. After a search operation, you can use the `S` command alone to search for the previously entered string. True or false?

_____ 27. You can search for and replace control characters. True or false?

_____ 28. You can use `^V` to insert in a batch file a control character that cannot otherwise be inserted. True or false?

_____ 29. The `1,#RSmith^Z` command will:
a. search for `Smith`
b. replace `Smith` with `^Z`
c. delete `Smith` in the current line
d. delete `Smith` in the entire file

_____ 30. The `T` command will allow you to transfer text lines from memory to disk. True or false?

Summary and Review

Check each item as you review and understand the following highlights of the chapter.

- ☐ 1. EDLIN, being readily available, is convenient for editing simple text files, such as batch files.
- ☐ 2. Before EDLIN can be loaded, you must specify a file name, even if the file does not exist.
- ☐ 3. Use the `I` (Insert) command in a new file to start typing text lines. This command can also be used to insert at the current line or a line specified before the command. Press Ctrl-C or Ctrl-Break to end insertion.
- ☐ 4. Enter a line number to edit that particular line, or a period (.) to edit the current line (marked with *), or just press Enter to edit the line after the current line.
- ☐ 5. Use the available DOS editing keys, such as F1, F2, F3, F4, and F5 to speed up editing.

- ☐ 6. Use the C (Copy) command to duplicate lines, M (Move) to move lines, and D (Delete) to delete lines.
- ☐ 7. Use the L (List) command to list a range of lines without changing the current line. Use the P (Page) command to page through the file and change the current line.
- ☐ 8. Use the S (Search) command to search for a line containing a specified string. Place ? before S to display the O.K.? prompt.
- ☐ 9. Use the R (Replace) command to search for a string and replace it with another. Separate the two with ^Z. This command can also be used to delete a string. It is advisable to use the command with the ? option of displaying a prompt. This lets you decide whether you really want to replace a string.
- ☐ 10. Control characters are often necessary in a batch file. If a control character cannot be inserted in your file by pressing Ctrl and a letter (or another character), you need to first insert Ctrl-V (^V) and then type that letter.
- ☐ 11. Use the E (Exit) command to save a file and exit EDLIN, or the Q (Quit) command to abandon the edited file and leave EDLIN.
- ☐ 12. When a new version is saved, the old file is renamed with the BAK extension. When the latest version is not accessible, rename the backup version before retrieving it.

Review Questions

Write an answer for each question below.

1. If you try to run EDLIN and get a `Bad command or file name` message, what is the reason?
2. What happens if you just enter EDLIN without a file name?
3. How do you know whether a file you are editing is a new one or an existing one?
4. If you wish to edit an old file and get a `New file` message, why does it happen and what can you do?
5. What does the command `1,#d` do?
6. If you want to insert a line at the location of the current line, what do you do?
7. What happens if you press Ctrl-C at the end of a line you are editing or inserting?
8. What is the quickest way to designate a line as the current line?
9. When editing a line, which DOS key lets you retrieve all the characters in the buffer (template)?
10. What happens if you just press Enter at the * command prompt?
11. How do the L and P commands differ?
12. If you use the S command without specifying a range of line numbers, how does EDLIN respond?
13. What happens if you just enter the S command after a search operation?

14. If you want EDLIN to display a prompt during a search or replace operation, what do you do?
15. If you want to enter a control character, such as ^ [, but EDLIN will not respond to your Ctrl-[combination, what can you do?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
BAK files	EDLIN can edit files with the BAK extension	EDLIN can't edit them; extension must be changed
L, P commands	Can display no more than 23 lines at one time; no pause for a long list of lines	Can display 24 lines; if more, will pause and display: Continue (Y/N)?

Programming Batch Files

Chapter 11

TOPICS:

Batch Commands
Controlling Screen Display (ECHO, PAUSE, @, REM, :)
Maneuvering Dummy Variables (SHIFT, %)
Looping (FOR..IN..DO)
Branching (IF, IF NOT, GOTO, :, ==)
Testing File Existence and Exit Codes (IF EXIST, IF NOT EXIST)
Chaining and Nesting (CALL, COMMAND/C)
Building a Menu

COMMANDS COVERED:

ECHO, PAUSE, REM, SHIFT, FOR..IN..DO, IF, GOTO, EXIST, CALL

NEW TERMS AND SYMBOLS:

debugging, @, :, %, variable, replaceable parameter (dummy variable), looping, label, branching, nesting, chaining, exit codes, errorlevel

REQUIRED MATERIALS:

1. Two practice disks
2. Access to EDLIN.COM
3. Ability to use EDLIN

Quick Start

✎ **For a summary of batch commands:**
See Table 11.1.

✎ **To hide all batch execution, put this in the first line:**
`@echo off`

✎ **To create an empty line on the screen:**
`echo/`

✎ **To add remarks in a batch file:**
Place remarks after `REM` or `::`.

✎ **To display message and pause batch execution:**
Place message after `ECHO` and add `PAUSE` in a new line.

✎ **To maneuver more than 10 parameters:**
Use `SHIFT` to shift to each following parameter.

✎ **To loop a command a number of times (`%` on command line, `%%` in batch file):**

```
A>for %x in (file?) do type %x
```

✎ **To do unconditional branching:**

```
goto LABEL
...
:LABEL
```

✎ **To test parameter existence and do conditional branching:**

```
if .==.%1 goto HELP
...
:HELP
```

✎ **To test file existence:**

```
if exist %1 echo %1 is here
```

✎ **To test an exit code of a command:**

```
if error evel 2 goto END
```

You have been exposed to batch files before. But you have not been introduced to the batch commands that can make batch files more intelligent and productive. This chapter explores these commands. If you can skillfully utilize them, you can produce powerful batch files that resemble simple computer programs.

Batch Commands

✎ **Batch commands are summarized here.**

About a dozen commands and symbols, as shown in Table 11.1, are used in programming batch files. `FOR..IN..DO` is often entered on the command line. Other commands can also be entered on the command line. They are, however, mostly meaningless unless put in batch files. All these batch commands are internal. There is no need to worry about where DOS can access them. Most of these commands are used to control the flow of batch files so that they can be structured like computer programs.

Table 11.1 Batch Commands

Command	Purpose
@	Hide a line during execution
:	Mark a label
%	Replaceable parameter (dummy variable)
==	Test equality and parameter existence
CALL	Execute another batch file and return
ECHO	Turn on/off display; show message
EXIST	Check existence of a file
FOR..IN..DO	Repeat execution of a command
GOTO	Go to a label
IF (IF NOT)	Decision making for conditional branching
PAUSE	Halt execution
REM	Remark
SHIFT	Shift to the next parameter

These programming commands are rather limited when compared to a full-fledged computer language like C. They allow you to maneuver your batch files only in certain ways. Nevertheless, this relatively simple programming device can be a productivity tool. While it is not as capable as a programming language, it is much easier to learn. When skillfully used, it can save time in automating many DOS tasks.

Controlling Screen Display (ECHO, PAUSE, @, REM, :)

- ✎ **Use ECHO to display a message on the screen.**
- ✎ **Use PAUSE to halt a batch file's execution.**
- ✎ **Use ECHO on/off to display/hide execution of commands.**
- ✎ **Put @ before a command to suppress its display.**
- ✎ **Use REM (or :) to add remarks to batch file lines.**

Three commands (ECHO, PAUSE, and REM) and one symbol (@) are used to provide remarks in batch files and control the screen display —pausing execution and displaying messages.

ECHO and PAUSE

Normally, a batch file is executed with the default of ECHO on. Every single line is displayed on the screen as it is executed. This arrangement is useful in **debugging** (identifying and eliminating errors) a batch file. Each command or error message, if any, is displayed on the screen, allowing you to spot errant lines. You can pause the screen display or echo the display to the printer to examine the error messages.

In a long batch file, you can selectively insert ECHO ON and ECHO OFF in crucial places to identify bugs. The trouble-free portions will not be displayed, and only those that have potential problems will be echoed to the screen for your view.

After a batch file is completely debugged and error free, you should turn off ECHO. Then the batch file's execution will not clutter the screen and distract the user.

Although execution of commands is not displayed when ECHO is off, messages are not suppressed. You can still use ECHO to display a message and use PAUSE to halt batch execution to wait for a user response. The following batch file[1] will display a two-line message and pause for a response from the keyboard:

```
1: echo off
2: echo Turn on the printer
3: pause
4: copy *.txt prn
```

After running this batch file, the screen will pause and show these two lines:

```
Turn on the printer
Press a key to continue . . .
```

The first line of the message comes from ECHO in line 2, and the second line comes from PAUSE.

[1] All the batch file listings in this and the next chapter simulate the way they appear in EDLIN. If you use COPY CON or a word processor to create them, do not use line numbers.

If you want to replace the message produced by PAUSE, you can do something like this:

```
1: echo off
2: echo Turn on the printer
3: echo Press a key to print or ^C to abort
4: pause > nul
5: copy *.txt prn
```

Lines 2 and 3 will produce a two-line message, and line 4 will only pause. The default message of PAUSE will be discarded (redirected to NUL).

Note: Although you can suppress PAUSE's built-in message, you cannot alter it by using the PAUSE command alone. For example, consider the following:

```
pause Turn on the printer
```

Your message will be ignored and the original message will be shown instead. If you redirect the original message like this:

```
pause Turn on the printer > nul
```

nothing will be shown during the pause.

Using @ to Hide Display

The above batch file will not display the execution of the commands in lines 2–5, but the execution of line 1 will still be displayed. To remedy this situation, DOS versions 3.3 and later allow you to use @ to suppress a line's execution. We can thus modify line 1 of our batch file this way:

```
1: @echo off
```

After this modification, only messages, if any, are displayed and all other actions are hidden.

The @ symbol can be put in front of any line to suppress its display. For example, if you want to hide only line 5, you can do this:

```
5: @copy *.txt prn
```

Displaying Empty Lines

Sometimes you want to insert empty lines to make your batch message more readable. There are a number of ways to do that. In versions 3.xx and 4, the simplest way is to put a slash (/) immediately after ECHO:

```
1: @echo off
2: echo/
3: echo Turn on the printer
4: echo Press a key to print or ^C to abort
5: echo/
6: pause > nul
7: copy *.txt prn
```

After this alteration, the message will be preceded and followed by an empty line.

Tips & Tricks

ECHO and Empty Lines

Different DOS versions have different ways of creating empty lines. If you use version 4, put one of the following characters immediately after ECHO, leaving no space between a sign and ECHO:

```
:  ;  ,  .  "  +  /  \  [  ]
```

If there is an intervening space, a sign will be echoed to the screen, instead of creating an empty line. If you use version 3.xx, take out the comma (,) and semicolon (;) from the above list.

Some versions will work with ASCII 0 (^@; press F7 to insert) and 255 (hold down Alt and type 255 on the numeric keypad); other versions will not work. Consistency has not been a virtue of DOS.

This book uses / immediately after ECHO to create an empty line. This will work with all 3.xx and 4.xx versions.

A batch file's ECHO ON/OFF command will not affect the state of ECHO on the command line. However, the opposite is not true. When you run a batch file, the command line ECHO should be kept in the default ON state. Chapter 12 will elaborate more on this.

Tips & Tricks

ECHO and Characters

ECHO and a message are normally separated by a space. You can also use one of these characters in lieu of a space:

```
:  ;  ,  .  "  +  /  \  [  ]  =  (tab)
```

ECHO understands the piping and redirection signs of ¦ , >, < and >> and will interpret them as taking certain actions. To display them on the screen, you need to put them in double quotes.

Using REM or : for Remarks

The REM command is placed at the beginning of each remark line to document a command above or below it. Such a remark can later remind you or others of what a tricky line is supposed to do. Without such remarks, you or others may be puzzled and have difficulty using or modifying a batch file.

DOS commands placed after REM are not executed. For example, you may have this batch line:

```
rem echo dir is most useful
```

Neither ECHO nor DIR is executed.

Remarks are not completely ignored during batch execution. If ECHO is on, entire remark lines (including the beginning REM) are displayed; if off, they are suppressed.

If you insert piping and redirection symbols (>, <, >>, and |) without enclosing them in double quotes, they will be interpreted as performing actions, not just displaying signs. For example, a batch line like this:

```
rem A is > B
```

will create a 0-byte file named B—regardless of ECHO's state. On the other hand, this line will do nothing:

```
rem A is ">" B
```

Instead of REM, you could place a colon (:) at the beginning of a line. Anything after the colon, including redirection and piping symbols without quotes, is completely ignored. They are also not displayed regardless of the state of ECHO.

A colon is also used to identify a label for branching. If you use it for remarks, make sure the first word after a colon is not identical to the label you want the batch file to branch to. (See below for discussions of label and branching.)

Practice 11-1

☐ 1. Use EDLIN to create the following batch file named DEMO.BAT; run it after exiting EDLIN.

```
1: rem DEMO.BAT
2: echo This batch file is to demonstrate
3: echo the use of ECHO, REM, PAUSE, @.
```

☐ 2. Add a line to DEMO.BAT to hide its entire execution.

☐ 3. Modify DEMO.BAT to pause display after the first line of the message.

☐ 4. Replace the above PAUSE message with your own.

☐ 5. Add an empty line before and after the first line of message.

Drill

_____ 1. All the batch commands are internal. True or false?

_____ 2. If you do not include any ECHO ON command in a batch file, all the commands being executed will be displayed. True or false?

_____ 3. When ECHO is off, remarks are displayed. True or false?

_____ 4. If @ECHO OFF appears in the second line of a batch file, which of the following is hidden?
   a. only the first line
   b. the first two lines
   c. all the lines after the second
   d. all the lines from the second

_____ 5. A command such as DIR placed in the same line after REM will be executed. True or false?

_____ 6. If ECHO is entered alone in a separate line in a batch file, it will
   a. display ECHO on/off
   b. create an empty line
   c. do nothing
   d. do various things depending on the version

_____ 7. The `pause Turn on the printer > nul` command in a batch file will pause execution and display
   a. `Turn on the printer`
   b. `Press any key to continue . . .`
   c. nothing
   d. will not pause or display anything

_____ 8. The batch command `echo/` will create an empty line. True or false?

Maneuvering Dummy Variables (SHIFT, %)

✎ **Use %# as a place holder for a parameter.**

✎ **Use SHIFT to shift to each successive parameter.**

Every programming language allows you to maneuver variables. A **variable** is a letter or text string that can be used to represent different values, which can be numbers or text strings. Variables make programs versatile and useful.

DOS provides something akin to variables. Officially they are called **replaceable parameters**. Unofficially some people call them variables and others refer to them as **dummy variables** (or dummy arguments). They are not true variables because you cannot maneuver them like those found in a programming language;

you cannot, for example, assign numeric values to them or make them do arithmetic calculations. These dummy variables, a term more suitable to their nature, are used exclusively to maneuver parameters entered on the command line.

The percent sign (%), followed by a decimal number, is used to represent a dummy variable. It is like a place holder reserving a place for a parameter (argument) that is expected from the command line. The first item entered on the command line (parameter 0, batch file name) is placed in %0, the second (parameter 1) in %1, and so on up to %9. Here is an example:

```
1: @echo off
2: echo bach file name: %0
3: echo parameter 1: %1
4: echo parameter 2: %2
```

Suppose you name this batch file NAME.BAT. You can now enter the following to demonstrate how dummy variables work:

```
A>Name Laurel Hardy
batch file name: Name
parameter 1: Laurel
parameter 2: Hardy
A>_
```

If you enter more parameters than there are available dummy variables, the extras are ignored and not displayed. If you enter fewer parameters, some dummy variables are not assigned any value:

```
A>Name Laurel
batch file name: Name
parameter 1: Laurel
parameter 2:
A>_
```

Normally you can use only 10 dummy variables, %0 to %9. You can, however, use the SHIFT command to maneuver more. Every time this command is encountered, the next parameter entered on the command line is assigned to the same dummy variable. This provides some degree of mobility and versatility resembling the use of a true variable. This phenomenon is demonstrated below:

```
1: @echo off
2: echo %1
3: shift
4: echo %1
5: shift
6: echo %1
```

Suppose you name the above NUMBER.BAT. If you enter the first line below, the three lines after that are displayed:

```
A>number one two three
one
two
three
A>_
```

Even though you use only one dummy variable, all the three parameters entered on the command line are echoed to the screen. Each time SHIFT is executed, DOS

moves to the next parameter. The same dummy variable then holds a new value, the next parameter.

Shifting parameters is a valuable technique that can be used to construct simple but powerful batch files. The following is an example:

```
1: @echo off
2: rem A2B.BAT--copy files from A to B
3: :AGAIN
4: copy a:%1 b:
5: shift
6: if not .==%1. goto AGAIN
```

Suppose you name this batch file `A2B.BAT`. You can use it to copy multiple files with one command. This single command will copy two files:

```
A>a2b file1.txt test2.doc
```

When the batch file is executed the first time, line 4 treats `%1` as `FILE1.TXT` and copies it. Line 5 shifts the parameter to the next file, namely `TEST2.DOC`. Line 6 tests the existence of another parameter and branches to line 3. Then the new file placed in `%1` is copied. When line 6 next tests false, the batch file ends. (`IF` and `GOTO` will be explained below.)

You can also use wild card characters to copy all the matching files, such as:

```
A>a2b file?.txt
```

Tips & Tricks

Shifting Parameters

If you are puzzled as to how dummy variables and parameters are affected by `SHIFT`, the batch file below named `PARADEMO.BAT` may help:

```
1: @echo off
2: echo %0; %1; %2; %3
3: shift
4: echo %0; %1; %2; %3
5: shift
6: echo %0; %1; %2; %3
7: shift
8: echo %0; %1; %2; %3
```

Running this batch file with the four parameters after it shows this result:

```
A>parademo 1 2 3 4
parademo; 1; 2; 3
1; 2; 3; 4
2; 3; 4;
3; 4; ;
A>_
```

In the first round, `%0` represents the batch file name. In the second, the same dummy variable has the value of 1, and so on.

Looping (FOR..IN..DO)

✎ **Use the FOR..IN..DO command in a batch file (or enter it on the command line) to repeat an action a number of times.**

In computer programming, the technique of making a program perform a task a number of times is called **looping**. DOS provides a simple looping command. It has the befuddling syntax of:

```
FOR %%variable IN (set) DO action %%variable
```

It performs a function similar to BASIC's FOR-NEXT command. In format, however, it resembles this C looping command:

```
for (i=1; i<=10; i++)
```

In the C command, the variable i is to begin with the value of 1, the looping is to stop when i reaches 10, and i is to increase its value by 1 after each loop. In DOS, a variable (another dummy) is any letter, which is placed after %%. The set is a list of files. And a DOS command is to follow DO. Here is an example:

```
1: @echo off
2: for %%x in (file1.txt, file2.txt) do copy %%x b:
```

The dummy variable of x (which can be any letter) is to successively hold each of the files after IN and enclosed in parentheses. It holds a different value each time the loop is repeated. When the command (COPY) after DO is executed, it will first copy the first file, then the second, and so on until all the files in the list are copied.

Wild cards can be used with this command, such as:

```
2: for %%x in (*.txt) do copy %%x b:
```

All the matching files found in the current directory will be copied to drive B.

Version Variances: If you use version 3.2 or later, you can also designate a directory path inside the set, such as:

```
2: for %%x in (c:\wp\work\*.txt) do copy %%x a:
```

Earlier versions will work only with the files in the current directory.

The FOR..IN..DO command can be entered on the command line, which is rather elaborate and thus often unnecessary. If you do that, you must use one %, such as:

```
A>for %x in (*.txt) do copy %x b:
```

A less complicated way of doing the same thing is this:

```
A>copy *.txt b:
```

Note: A single % will work on the command line, but it will not work in a batch file; you must use %% in a batch file. In a batch file, if you use only one % with a dummy variable in the FOR command, DOS will give you a Syntax error message.

The looping command is most useful with commands such as TYPE that do not allow wild card characters. If you want to display the contents of all the matching files one after another, try this:

```
A>for %x in (file?.txt) do type %x
```

The looping technique can be useful in comparing files on two disks, such as:

```
1: @echo off
2: for %%x in (*.*) do if not exist b:%%x echo %%x
```

Suppose you name it COMPFIL.BAT and run it when drive A is the default:

```
A>compfil
```

The screen will show all the files found in drive A but not in drive B. This lets you quickly find out the discrepancies between two disks or directories—the current directory in the default drive and the current directory in drive B.

If you frequently need to back up files, to create duplicate files for safekeeping, the above batch file could be modified to include copying as well. The batch file below will do it:

```
1: @echo off
2: echo These files exist in source but not target:
3: for %%x in (c:\bat\*.bat) do if not exist a:%%x echo %%x
4: echo Press a key to copy, or ^C to abort.
5: pause > nul
6: for %%x in (c:\bat\*.bat) do if not exist a:%%x copy %%x a:
```

After displaying the discrepancies, the batch file pauses and awaits your response. If you press any key other than Ctrl-C (or Ctrl-Break), copying begins.

Warning: Use the DEL and FOR..IN..DO combination with great caution, such as this:

```
A>for %x in (*.*) do del %x
```

This command will delete all the files in drive A's current directory without warning. Since the command works on one file at a time, it does not trigger the Y/N prompt as this command does:

```
A>del *.*
```

Practice 11-2

1. ☐ Write a batch file that will echo to the screen all the parameters entered on the command line. Use a single SHIFT command in the batch file.
2. ☐ Modify the COMPFIL.BAT file discussed in the text so that it will display all the files found in both drives A and B.
3. ☐ Modify SHOWBACK.BAT to allow the source file pattern and target drive to be controlled by parameters entered on the command line.

Drill

_____ 9. Dummy variables (replaceable parameters) can be used to maneuver (add, subtract, etc.) numeric values. True or false?

_____ 10. Suppose DONE.BAT is a batch file. In the following command, DONE is treated as %0. True or false?

```
A>done file1 file2 file3
```

The batch file below, named COPYFIL.BAT, is used to answer questions 11–12:

```
1: copy %1 b:
2: copy %2 b:
3: copy %3 b:
```

_____ 11. The following command will produce an error message: 0 File(s) copied. True or false?

```
A>copyfil file1 file2
```

_____ 12. The following command will copy all the entered files. True or false?

```
A>copyfil file1 file2 file3 file4
```

_____ 13. Without using SHIFT, you cannot use more than 10 parameters in a batch file. True or false?

_____ 14. Assume TEST.BAT has these lines:

```
1: echo %1
2: shift
3: echo %1
```

The command below will cause line 3 to display the message line 2. True or false?

```
A>test line 1 line 2
```

_____ 15. The following command will:

```
A>for %n in (*.txt) do type %n
```

a. produce an error message
b. display contents of all matching files in A
c. display n number of files
d. display matching files in drive B

_____ 16. The above command line, without any change, is legal if put in a batch file. True or false?

_____ 17. The following command will produce a warning and display a Y/N prompt. True or false?

```
A>for %x in (*.*) do del %x
```

Branching (IF/IF NOT, GOTO, :, ==)

- **DOS recognizes a label as a new line beginning with a colon, followed by a text string.**
- **Use GOTO followed by a label to branch (unconditionally) batch execution to a label.**
- **Use IF (or IF NOT) and GOTO to do conditional branching.**
- **Use the common equation of `if .==%1. goto . . .` to test the existence of a parameter entered on the command line and do conditional branching.**

A batch file, like a computer program, normally runs from top to bottom. Each line is executed sequentially. You can, however, make it run more intelligently by using branching, chaining, and nesting. Branching is explained in this section; chaining and nesting will be covered later.

There are two kinds of **branching**, conditional and unconditional. Unconditional branching may resemble this scenario: Execute line 5 and then continue in line 10. Conditional branching could be: Execute line 1 and, if the user enters no parameter on the command line, go to line 20.

Labels are used to mark lines to guide batch execution.[2] A **label** is simply a text string preceded by a colon (:), which is placed at the beginning of a line. It is nothing more than a place marker, marking a particular place where DOS is to branch when instructed to do so.

DOS uses the `IF` command to do conditional branching. Compared to other programming languages' counterparts, DOS's `IF` command is very limited. It can be used to test only three conditions:

1. Existence of a parameter (command-line argument)
2. Existence of a file on disk
3. Exit codes of certain DOS commands

These are covered in the following sections.

Displaying and Deleting Files

To test whether a condition exists, use `IF` alone. To test whether a condition does not exist, use `IF NOT`. The following batch file uses `IF` to test the existence of a parameter:

```
1: @echo off
2: rem KILL.BAT--display files before mass deletion
3: rem if no parameter (filename) is entered
4: if .==.%1 goto HELP
5: dir %1
```

[2] Unlike in BASIC, a batch file does not use line numbers.

Tips & Tricks

The Insensitive Label

Labels are not case sensitive and can be in uppercase or lowercase. However, if you have two labels of identical spelling but not in the same case, only the first can be reached, such as the following:

```
1: goto PLATO
2: echo test plato
3: :plato
4: echo 1st label
5: goto end
6: :PLATO
7: echo 2nd label
8: :END
```

Line 1 leads to line 3 (not line 6), and then line 5 leads to line 8. Line 2 is skipped, and lines 6 and 7 cannot be reached because another label of the same spelling precedes it. To make labels more readable, using uppercase consistently is a good idea.

```
 6: echo/
 7: echo Press a key to delete these files, or ^C to abort.
 8: pause > nul
 9: del %1
10: goto END
11: :HELP
12: echo/
13: echo You must enter a parameter, such as: %0 *.txt
14: echo/
15: :END
```

This batch file is intended to provide a safeguard when you use wild card characters to delete files. It displays all the matching files slated for deletion and pauses for you to press Ctrl-C to abort or another key to begin deletion.

Line 4 tests whether or not a parameter has been entered on the command line. If no parameter is entered, in other words only the batch file name is entered, the equation in line 4 becomes:

```
.==.
```

That means the `IF` statement is true. Execution then branches to the `HELP` label in line 11. The message in line 13 is displayed like this:

```
You must enter a parameter, such as: kill *.txt
```

In addition to the regular text, line 13 also converts `%0` to `kill`, the name of the batch file.

After the message is displayed, the batch file runs out of lines to execute and stops. (Keep in mind that a label like `:END` does nothing except mark a place for branching.)

If line 4 does not test true, namely, a parameter is entered on the command line, the action after the equation is not taken. Instead, the line immediately below is executed.

Line 5 displays all the files matching the pattern entered on the command line. Line 7 displays a message, and line 8 pauses batch execution. If you press any key other than Ctrl-C or Ctrl-Break, the displayed files are deleted.

Line 10 branches to line 15, where the batch file ends its execution as there is nothing else to execute. Line 10 is an example of unconditional branching. No test is made; the flow of execution is simply directed to a label. If line 10 did not exist, lines 11–14 would be executed, displaying an unnecessary and misleading message.

Tips & Tricks

Parameter Testing

A wide variety of expressions are used to test the existence of parameters entered on the command line. The variations shown below are all legal:

```
if !==!%1 goto HELP
if x==%2x goto OOPS
if ""=="%3" goto NONE
if .%1==.lotus goto LOTUS
```

The double equal sign, instead of a single one, is used to test equality, as in C but not in BASIC or Pascal.

There must be one (the same) dummy character (any character) before and after ==. If not, an error message will be generated when no parameter is entered on the command line.

The decimal number after the % sign depends on the number of parameters you expect the user to enter. For example, if you expect three parameters after the batch file name, use `%3`.

Moving Files

A slightly different arrangement can be used to move files from one directory to another:

```
1: @echo off
2: rem MOVE.BAT--move files from one dir to another
3: if .==%1. goto HELP
4: if .==%2. goto HELP
5: dir %1 /p
6: echo/
7: echo These files are to be moved to %2
8: pause
9: copy /v %1 %2
10: echo/
11: echo Press a key to delete the original files.
12: pause > nul
13: del %1
14: goto END
15: :HELP
16: echo/
17: echo Enter filename and target drive, such as:
18: echo       %0 file? b:
19: :END
```

Lines 3 and 4 test to see whether two parameters follow the batch file name. If not, execution branches to the HELP label, where an example is provided and batch execution ends.

If at least two parameters are entered after the batch file name, the matching files to be moved are displayed (line 5). After you press a key, line 9 copies them (with Verify on) to the target. After another pause, line 13 deletes all the matching files.

You can use wild card characters, such as:

```
A>move file? b:
```

This moves the matching files from A to B. If you want to move from B to A, try this:

```
A>move b:file? a:
```

You need to specify both the source and target, or the HELP message will show up.

Printer Enhancement

The opposite of IF-GOTO is IF NOT-GOTO. It can be used to test the nonexistence of a condition. We will use it to construct a routine to let you enhance your dot-matrix printer:

```
1: @echo off
2: rem EPSON.BAT--set various print enhancements
3: if not .==%1. goto %1
4: echo Options:
5: echo 1 = NLQ  2 = Compressed  3 = Expanded
6: echo 4 = Elite  5 = Double strike  6 = Reset
7: goto END
8: :1
9: echo ^[x1 > prn
```

```
10: goto END
11: :2
12: echo ^O > prn
13: goto END
14: :3
15: echo ^[W1 > prn
16: goto END
17: :4
18: echo ^[M > prn
19: goto END
20: :5
21: echo ^[G > prn
22: goto END
23: :6
24: echo ^[x0^R^[W0^[P^[H > prn
25: :END
26: shift
27: if not .==%1. goto %1
```

Two tests are conducted, in lines 3 and 27. If no parameter is entered, line 3 tests false and lines 4–7 are executed, displaying the available options. So, if you just enter the batch file name alone, you get the instructions:

```
A>epson
1 = NLQ  2 = Compressed  3 = Expanded
4 = Elite  5 = Double strike  6 = Reset
```

If a parameter is entered, such as:

```
A>epson 1
```

line 3 tests true, batch execution branches to a particular label matching the parameter, and your printer is set to Near Letter Quality mode. Make sure your printer is turned on. If a control code is registered, the roller will move up by one line.

If you enter multiple parameters, such as:

```
A>epson 3 5
```

line 3 will branch execution to line 14 to set the expanded mode. Then line 27, detecting the existence of another parameter, will branch batch execution back to line 20 to activate double strike.

Warning: Keep in mind that some of these control codes are not compatible with one another. For example, if you enter 1 and 3, only 1 will be accepted and 3 will be ignored. You may also need to cancel all the existing codes by specifying the 6 parameter before you enter another one.

These codes are valid with an Epson dot-matrix printer. You are free to add other options such as italic and superscript. Your printer may have different control codes; if so, use those control codes instead.

Password Protection

In Chapter 9 we discussed environment variables. We are going to use one to provide for a password. The batch file below uses `IF-GOTO` to test the existence of a parameter and `IF NOT-GOTO` to test whether the password entered on the command line is equal to the one in the environment:

```
 1: @echo off
 2: rem PASS.BAT--demo batch use of environment
 3: if .==%1. goto NOPAR
 4: if not .%password%==.%1 goto NOGO
 5: echo You have entered the correct password.
 6: rem Enter your program here.
 7: rem . . . .
 8: goto END
 9: :NOGO
10: echo You have failed to entered the correct password.
11: goto END
12: :NOPAR
13: echo You have not entered a password.
14: echo To use this program, you have to follow this:
15: echo       %0 password
16: :END
```

Before you run this batch file, you need to create an environment variable and assign a value to it, such as:

```
A>set password=tora
```

If a user now runs the batch file without a parameter, line 3 branches execution to line 12 and this error message is displayed:

```
A>pass
You have not entered a password.
To use this program, you have to follow this:
     pass password
```

If a parameter is entered, line 4 tests to see whether it matches the one in the environment. If not, this is the result:

```
A>pass nincompoop
You have not entered the correct password.
```

If the two match, line 5 is executed and your program is run.

A savvy user can use the SET command to see the password. To foil such a person, try this command:

```
A>set password=<Alt-255><Alt-255>
```

After the = sign, you are to hold down Alt and type 255 on the numeric keypad; repeat it again.[3] This enters ASCII 255 (blank character) twice. If you enter the SET command alone, you see only this:

```
PASSWORD=
```

Nothing is displayed after the = sign. To enter a correct password after this move, you need to do this:

```
A>pass <Alt-255><Alt-255>
```

That means to enter ASCII 255 twice after typing the batch file name and pressing the space bar once; nothing else will work.

[3] If your keyboard does not respond to Alt-255, see footnote 3 in Chapter 4 for a remedy.

To use the batch file to automatically enter the password into the environment, add this line between lines 2 and 3:

```
set password=<Alt-255><Alt-255>
```

If you now make your batch file invisible (hidden) by using a simple assembly program, which is available in many computer bulletin boards or bundled with some MS-DOS versions as CHMOD or CHMODE, you have a largely foolproof system.

Practice 11-3

☐ 1. Create a batch file that will allow a user to enter a file name and a person's name (or number) on the command line to display records (lines) found. Use it on the telephone database created in the previous chapter. Use the FIND command to do the search. Provide branching to a help line if two parameters are not entered.

☐ 2. Modify MOVE.BAT to allow for multiple files with dissimilar names to be moved with a single command from the current directory to a target entered on the command line. For example, this command will move to drive A all the listed files:

```
C:\WP\REPORT>move a: report.90 budget.txt propose.doc
```

(Hint: Use %1 for the target drive/directory and make it an environment variable to freeze it so it cannot be shifted. Use %2 to hold the file list and SHIFT to change its value. Use IF to test the existence of another parameter for looping.)

☐ 3. Modify EPSON.BAT in the text to include an option for italics. (If your printer uses different codes, create a batch file like EPSON.BAT that can control the printer.)

☐ 4. In PASS.BAT, change IF to IF NOT and IF NOT to IF. (Hint: Use EPSON.BAT as a model for rearranging batch lines.)

Drill

_____ 18. If there are three lines marked by the three labels :label, :Label, and :LABEL, the goto Label batch command will branch to:
a. whichever is located closest to the command
b. :Label, because of exact matching in case
c. :LABEL, because a label must be in uppercase
d. the one located nearest the top of the batch file

_____ 19. A batch file can have line numbers like those found in a BASIC program. True or false?

_____ 20. The command `if m==%3m goto HELP` means:
   a. no more than three parameters are expected
   b. branch to `HELP` if parameter `%3` is absent
   c. both 1 and 2
   d. is a meaningless test

Testing File Existence and Exit Codes (IF EXIST, IF NOT EXIST)

- ✎ **Use `IF` or `IF NOT` to test the existence of a file on disk before taking an action with it.**
- ✎ **Use `IF` or `IF NOT` to test an exit code of some DOS commands and branch batch execution accordingly.**

The `IF` command can test three items, as explained in the preceding section. Testing the existence of a parameter was explained there. This section explains file existence and exit codes.

File Existence and Copying

`IF` is used with `EXIST` or `NOT EXIST` to test the existence of files in any drive or directory. The `DUPE.BAT` batch file below tests to see if the target drive or directory has duplicate files. If there are no duplicates, copying begins immediately. If duplicates exist, they are displayed and action is halted. You can then decide whether to go ahead with copying or break the batch execution.

```
 1: @echo off
 2: rem DUPE.BAT--warn before overwriting targets
 3: if .==%1. goto HELP
 4: if .==%2. goto HELP
 5: if exist %2\%1 goto WARN
 6: :DO
 7: copy %1 %2
 8: goto END
 9: :WARN
10: echo These files exist in target drive:
11: dir %2\%1 /p
12: echo/
13: echo Press a key to copy, or ^C to abort.
14: pause > nul
15: goto DO
16: :HELP
17: echo You must enter two parameters, such as:
18: echo/
19: echo %0 a:file? b:
20: echo %0 *.txt b:\myfile
21: echo/
22: :END
```

The IF command first tests to see whether two parameters follow the batch file name. If not, the HELP message is shown, where an example is provided.

When correct parameters are entered, they are used by IF to check the existence of duplicate files in the target. Suppose you enter this command:

```
A>dupe *.bat b:\bat
```

The dummy variables in line 5 will treat your entry as:

```
b:\bat\*.bat
```

If matching files are found to exist, execution branches to the WARN label in line 9. Line 10 echoes the message, and line 11 executes the DIR command with this parameter:

```
dir b:\bat\*.bat /p
```

Line 14 pauses the action. If you press a key other than Ctrl-C or Ctrl-Break, execution branches to the DO label in line 6. Line 7 then performs this command:

```
copy *.bat b:\bat
```

After copying, execution branches to the END label and the batch file ends.

Exit Codes and File Backup

Some DOS commands report **exit codes**, some of which are shown in Table 11.2. These are the decimal numbers indicating different conditions of execution. These numbers can be tested by IF for branching.

The IF command can be used with `if errorlevel` to test each exit code by checking whether it is greater than or equal to an **error level**. The labels below can be in any order. However, you need to arrange the testing lines from higher to lower numbers, as shown below:

```
 1: @echo off
 2: rem BACK.BAT--back up all files in current dir
 3: path=c:\dos
 4: backup *.* a:
 5: if errorlevel 4 goto EL4
 6: if errorlevel 3 goto EL3
 7: if errorlevel 2 goto EL2
 8: if errorlevel 1 goto EL1
 9: goto END
10: :EL1
11: echo No source file is found
12: goto END
13: :EL2
14: echo Some files not backed up due to sharing
15: goto END
16: :EL3
17: echo Aborted by ^C
18: goto END
19: :EL4
20: echo Aborted due to error
21: :END
```

Table 11.2 Exit Codes

Command	#	Meaning
BACKUP	0	Successful action
	1	Source files not found
	2	Some files not backed up due to sharing
	3	Aborted due to ^C
	4	Aborted due to error
FORMAT	0	Successful action
	3	Aborted due to ^C
	4	Aborted due to error
	5	Aborted due to N response for hard disk
XCOPY	0	Successful action
	1	Source files not found
	2	Aborted due to ^C
	4	Assorted initiation error
	5	Aborted by user due to read/write error

(Note: XCOPY no longer reports exit codes in version 4. Other commands reporting exit codes include: DISKCOMP, DISKCOPY, GRAFTABL, KEYB, REPLACE, and RESTORE.)

This batch file enters (line 3) C:\DOS into the environment so DOS can go there to load BACKUP.COM and access FORMAT.COM. It then backs up all the files from the current directory to drive A. If the task is successfully implemented, the batch file ends without showing any message. If not, one of the four messages is displayed.

If you want to add modified files, you may want to create another batch file like the one below:[4]

```
 1: @echo off
 2: rem AUTOBACK.BAT--add modified files to backup disk
 3: echo/
 4: echo Add modified files to backup disk in drive A
 5: echo Insert the backup disk in drive A and
 6: pause
 7: :AGAIN
 8: path=c:\dos;c:\bat
 9: backup %1 a: /m/a < cr
10: if errorlevel 2 goto END
```

[4] This batch file is a modified version of the one discussed in Chapter 7.

```
11: if errorlevel 1 echo %1 not exist or modified
12: shift
13: if not x==%1x goto AGAIN
14: :END
```

This batch file allows you to enter varied file names on the command line to back up among them only those modified since last backup. You can enter a command like this:

```
C:\WP\WORK>autoback letter.* sales.pit report.90
```

Each of these files will be backed up if it has been modified. If a file in the list is not modified or is nonexistent in the current directory, line 11 tells you so. Line 12 shifts to the next parameter and line 13 tests to see whether there is another file. If so, it is tried again; if not, the batch file ends.

If an error other than 1 (no source file) is encountered, line 10 simply branches to line 14 and the batch file ends without any more action.

Line 9 redirects an input file named CR. It consists of a single empty line. You can create it this way:

```
C:\WP\BAT>copy con cr

^Z
```

This file provides the pressing of a key when BACKUP requires it. With this input redirected from an existing file, AUTOBACK.BAT works in a highly automated manner, requiring no further action from you after the pause in line 6.

Note: Beware of the difference between "if errorlevel" and "if not errorlevel." The former tests exit codes in descending order and the latter in ascending order. For example, "if errorlevel 5" tests true if the expected exit code is 5 or higher. This is comparable to this C language expression:

```
if (x >= 5)
```

On the other hand, "if not errorlevel 5" tests true only if the expected exit code is less than 5. It is thus comparable to this C expression:

```
if (x < 5)
```

Under some circumstances, you may need to combine both expressions, such as:

```
if errorlevel 5 if not errorlevel 6 goto . . .
```

The first expression will match 5 or higher, and the second any number lower than 6. As a result, only 5 and nothing else will match. See the MENU.BAT file below for an example of using this combination.

Practice 11-4

☐ 1. Create a batch file that will format a diskette and report various unsuccessful circumstances.

☐ 2. Create a batch file that tests existence of a file (whose name is entered on the command line) in two drives (A and B). If it exists in both, use `COMP` or `FC` to compare them.

☐ 3. Enter two commands on the command line, one to test whether a file exists in the current directory and another command to test a file's existence in another directory. Do not use `DIR`.

☐ 4. Modify `AUTOBACK.BAT` so that all error levels will be reported.

Drill

Answers for questions 21–24: a. `if not errorlevel`, b. `if not x==%x`, c. `if not exist`.

_____ 21. Test existence of a parameter.

_____ 22. Test existence of a file.

_____ 23. Test exit codes.

_____ 24. Used with a limited number of DOS commands.

_____ 25. When `if errorlevel` is used, the lines testing error levels should be arranged in a reverse order, with higher numbers placed before lower numbers. True or false?

Chaining and Nesting (`CALL`, `COMMAND/C`)

- **Use `CALL` in a batch file to execute another batch file and return.**
- **Execute another batch file from within a batch file to chain the new batch file.**

Among the common programming techniques, we previously discussed branching. This section explains the other two techniques of chaining and nesting.

Chaining vs. Nesting

The two techniques work differently. **Chaining** transfers control to another batch file. **Nesting** "calls" another batch file (or program). Chaining may or may not return control to the original batch file, depending on what the new batch file does. Nesting always returns control to the original (calling) batch file.

When you run a computer program, including a DOS external command, from inside a batch file, the program is nested. When you run another batch file (instead of a program), however, it is chained. Consider this example:

```
1: rem nesting
2: cd\bat
3: edlin x.bat
4: cd\
```

This batch file first changes the directory to \BAT and runs EDLIN with the file of X.BAT. Upon exiting EDLIN, line 4 is executed and the root directory is restored.

If you run another batch file, the outcome will be quite different. Suppose you have 2ND.BAT as shown below:

```
1: rem 2ND.BAT
2: echo 2nd batch, the only line
```

Suppose you now run 1ST.BAT as shown below:

```
1: @echo off
2: rem 1ST.BAT chaining 2ND.BAT
3: echo 1st batch, line 1
4: 2nd
5: echo 1st batch, line 2
```

The result below may surprise you:

```
A>1st
1st batch, line 1
2nd batch, the only line
A>_
```

By running 2ND.BAT in line 4, control is transferred to 2ND.BAT. Line 5 is thus not executed.

If you want 1ST.BAT to nest (not chain) 2ND.BAT, you need to change line 4 to this:

```
4: call 2nd
```

If you run the modified 1ST.BAT, this is the result:

```
A>1st
1st batch, line 1
2nd batch, the only line
1st batch, line 2
A>_
```

When CALL is used, 1ST.BAT nests 2ND.BAT. After 2ND.BAT is finished, control is returned to 1ST.BAT.[5]

Nesting is a useful technique in a complicated batch file. Smaller routines can be stored in separate batch files and called by the main batch file. This makes it easier to organize a complex batch file. Another benefit is that some of the small batch files can be recycled (called by several batch files), thus saving programming time.

[5] The CALL command is akin to a subroutine call (GOSUB–RETURN) of BASIC or a function call of C. The only difference is that in a batch file the called (nested) subroutine is outside the calling batch file.

Version Variances: The CALL command is not available before version 3.3. If you have an earlier version, you need to substitute it with COMMAND/C. DOS will load another copy of COMMAND.COM. Thus you need to provide a path to the command and the whole process can take some time. Users of 3.3 and later versions can still use this technique, but it is not necessary.

Quick Return to the Original Directory

Now let us explore a couple of practical batch files in which one can call the other. If you work through a complex directory system, you may wish to be able to do something in another directory and then quickly return to the original directory. The batch files discussed here may be what you need.

These two batch files assume that you use C:\BAT to store all your batch files and that you have in the environment a command like PATH=C:\BAT to enable you to run your batch files wherever you are.

```
1: @echo off
2: rem A.BAT, to create B.BAT
3: rem copy CD and a space to B.BAT
4: type c:\bat\space > c:\bat\b.bat
5: rem append current dir info to B.BAT
6: cd >> c:\bat\b.bat
7: cd\
8: rem append default drive letter to B.BAT
9: cd >> c:\bat\b.bat
10:
11: rem . . .
12:
13: call b
```

A.BAT, shown above, is the main batch file. It needs an existing file named SPACE, which has one single line. Use COPY CON to create it:

```
C:\BAT>copy con space
cd<space>
^Z
```

Note: Make sure to press the space bar once after CD. This space is needed to separate commands that will come in later.

A.BAT (think of it as Advance To) will create B.BAT (think of it as Back To). It will first create B.BAT and copy (line 4) to it the keystrokes in the SPACE file. Then it will append (line 6) the current directory data to B.BAT. Finally it will append (line 9) the default drive letter to B.BAT.

After line 9 you can add your routine to do whatever you want. When the routine is finished, line 13 calls B.BAT, which will take you to the original directory. The CALL command in line 13 can be omitted; in either case, control will be transferred to B.BAT.

If you do not want to add any routine, just end A.BAT in line 9. In this arrangement, running A.BAT creates B.BAT and puts you in the root directory. You can now do whatever you want. When you are ready to return, just run B.BAT.

If you are in C:\DOS when you run A.BAT, the first 9 lines will create B.BAT, containing these lines:

```
@echo off
cd C:\DOS
C:\
```

The first line is automatically transferred from A.BAT's first line. The second line comes from lines 4 and 6, and the third line from line 9 of A.BAT. These last two lines provide the commands that take you back to the original directory where A.BAT was executed.

Drill

_____ 26. After a batch file runs an external program, exiting the program always returns you to the DOS prompt. True or false?

_____ 27. If you have A.BAT and B.BAT, line 2 of the following batch file will never be executed. True or false?

```
1: a
2: call b
```

_____ 28. If batch file A calls (using CALL) batch file B, control returns to A after B runs its course. True or false?

_____ 29. Both CALL and COMMAND/C will run another copy of COMMAND.COM. True of false?

_____ 30. If the following batch file is named X.BAT, it will run indefinitely. True or false?

```
1: echo one
2: x
```

Building a Menu

- **Use a simple assembly program to provide an interactive mechanism and incorporate it into a menu batch file to make it more automated.**
- **Draw a double-line box around a menu display to make it more attractive.**

For our final project in this chapter, we are going to build a menu that will display the available options and allow you to press a single key to start a program. The

mechanism of pressing a single key to start an action is not available in the regular DOS commands. We need to enlist a tool used mostly by computer programmers, a mini-assembler called DEBUG.COM. It is available in the DOS package. We will use it to create an interactive mechanism. (You do not need to understand DEBUG; just follow the procedure below.)

Insert the disk containing DEBUG.COM in drive A (or anywhere as long as DOS can access it) and follow these keystrokes:

```
A>debug
-a
26FB:0100 mov ah,1
26FB:0102 int 21
26FB:0104 mov ah,4c
26FB:0106 int 21
26FB:0108
-rcx
CX 0000
:8
-n key.com
-w
Writing 0008 bytes
-q
```

Actually, what you really need to type, after DEBUG is loaded, are the following keystrokes:

```
a
mov ah,1
int 21
mov ah,4c
int 21

rcx
8
n key.com
w
q
```

Everything else not shown above is automatically entered by DEBUG. This creates an assembly program that is 8 bytes long and named KEY.COM. Check your disk to see its existence.

Note: If you do not want to go to the DEBUG screen, use COPY CON to create the above file and name it KEY.SCR (or any name) and then enter this command:

```
A>debug < key.scr
```

This will redirect the input file and automatically create KEY.COM without your having to go to the DEBUG screen.

Once this program exists, you can use it in a batch file to provide for the mechanism of pressing a key to get immediate action. The KEY.COM program is incorporated in line 5 of the menu batch file below:

```
 1:*@echo off
 2: rem MENU.BAT--show available options
 3: cls
 4: echo Select: W (WordPerfect), L (Lotus), D (dBASE)
 5: key
 6: echo/
 7: if errorlevel 119 if not errorlevel 120 goto WP
 8: if errorlevel 108 if not errorlevel 109 goto LOTUS
 9: if errorlevel 100 if not errorlevel 101 goto DBASE
10: if errorlevel 87 if not errorlevel 88 goto WP
11: if errorlevel 76 if not errorlevel 77 goto LOTUS
12: if errorlevel 68 if not errorlevel 69 goto DBASE
13: goto NONE
14: :WP
15: wp
16: :LOTUS
17: 123
18: :DBASE
19: db
20: :NONE
21: echo Not a valid selection
```

Line 4 displays the available options. Line 5 runs the KEY.COM program. This program pauses the batch file and waits for a keystroke. When a key is pressed, it is displayed on the screen and the real action begins!

The key pressed from the keyboard is now checked for its exit code. Each letter has an exit code, which is its ASCII value. The letter w, for example, has the ASCII value of 119. So line 7 intercepts the keypress and branches execution to the WP label in line 14. Line 15 then runs a batch file named WP.BAT. The MENU.BAT is finished, and control is transferred to WP.BAT.

WP.BAT may contain something like the following, depending on where you store your files:

```
1: @echo off
2: cd\wp\work
3: \wp\wp
4: cd\
```

In this arrangement, C:\WP\WORK is designated as the current directory and WP.EXE is started from the C:\WP directory. When you exit WordPerfect, line 4 changes back to the root directory. Instead of storing them in a separate batch file, these lines can be inserted in MENU.BAT.

Lines 7–12 of MENU.BAT require a little elaboration. Each of these lines is intended to isolate a particular letter and branch to a proper line. The first three lines intercept lowercase w (119), l (108), and d (100); and the last three lines, uppercase W (87), L (76), and D (68). When any one of these letters is pressed, execution is branched to a proper line. When any other letter is pressed, execution will branch to NONE and the batch file will end, after displaying the message in line 21.

Line 4 of MENU.BAT displays a one-line menu at the top of a clear screen. If you want to have a more attractive menu, first change it to:

```
4: type menu.txt
```

Then set out to create another file named MENU.TXT. This file could be as simple as this:

```
 1:
 2:
 3:
 4:      ╔═══════════════════════════════════════════════╗
 5:      ¦                                               ¦
 6:      ¦                                               ¦
 7:      ¦              Press A Letter:                  ¦
 8:      ¦                                               ¦
 9:      ¦                                               ¦
10:      ¦              D - dBASE IV                     ¦
11:      ¦                                               ¦
12:      ¦              L - Lotus 1-2-3                  ¦
13:      ¦                                               ¦
14:      ¦              W - WordPerfect                  ¦
15:      ¦                                               ¦
16:      ¦                                               ¦
17:      ¦                                               ¦
18:      ╚═══════════════════════════════════════════════╝
```

When you use the TYPE command to show this file, the contents will be displayed in the middle of the screen. It is not particularly attractive. Let us change it to a double-line rectangle by following these steps:

1. Go to line 4. With the cursor at the first =, hold down Alt and type 201 on the numeric keypad.
2. Press F3 and then the Left Arrow to locate the cursor at the end of the line. Enter Alt-187.
3. Go to the beginning of line 18 and enter Alt-200.
4. Go to the end of line 18 and enter Alt-188.
5. Replace all = with ═(Alt-205) with this command:

   ```
   *1r=^Z═
   ```

6. Replace all ! with ||(Alt-186) with this command:

   ```
   *1r!^Z||
   ```

The graphic signs and their ASCII values are shown in the ASCII table in Appendix B.

The result is an attractive menu shown below:

```
 1:
 2:
 3:
 4:
 5:
 6:
 7:
 8:
 9:
10:
11:
12:
13:
14:
15:
16:
17:
18:
```

```
Press A Letter:

D - dBASE IV

L - Lotus 1-2-3

W - WordPerfect
```

From now on, whenever you run MENU.BAT, this screen will greet you. If you want it displayed as soon as DOS is booted, change MENU.BAT to AUTOEXEC.BAT and store it (as well as MENU.TXT and KEY.COM) in the root directory of the boot disk.

Note: The KEY.COM file is included on the DOS-TA disk. It is used by the MENUB.BAT batch file, also on the disk. This batch file is a more complex version of the menu batch file discussed above. It can be used to run the Drill lesson files. It runs slower than the MENU.EXE program, also included on the disk. If you wish, you can retrieve MENUB.BAT into EDLIN and modify it for your own purposes.

Practice 11-5

☐ 1. Change the first item (dBASE IV) in MENU.TXT to E - EDLIN. Set it up in such a way that when a user presses E (or e), EDLIN will be loaded and a file named TEST.BAT will be either created or retrieved. Modify the entire menu structure accordingly.

☐ 2. Change the double-line border of MENU.TXT to a single-line border. Consult Appendix B for the proper characters.

☐ 3. Add a Quit option to MENU.BAT. The user should be allowed to press Q to end the batch file. If an unavailable option is chosen, the menu is to be displayed again.

☐ 4. Create a batch file that will provide help for three DOS commands each in a separate screen; the text for each help screen should be stored in a separate file. The batch file should provide for these scenarios:

1. No parameter entered
2. Available option
3. Unavailable option

Use the `DOS.EXE` program included on the DOS-TA disk as a model.

Summary and Review

Check each item as you review and understand the following highlights of the chapter:

☐ 1. There are about a dozen batch commands and symbols that can be used in batch files to make them behave similar to computer programs. Use these commands in batch files to make them more intelligent and to make you more productive.

☐ 2. Keep the default `ECHO` on when debugging a batch file. Turn it off after it is debugged. Put the `@ECHO OFF` command in the first line to hide execution of the entire batch file. Or put `@` in front of selected lines to hide their execution.

☐ 3. Use `REM` (or `:`) to add remarks. They can remind you of what some tricky lines are supposed to do. They are useful if you need to revise a batch file.

☐ 4. Dummy variables (replaceable parameters) allow you to handle parameters entered on the command line. There can be up to 10 such variables at one time.

☐ 5. Use the `SHIFT` command to handle more than 10 parameters. The command can make dummy variables more mobile and versatile.

☐ 6. `GOTO` and `:` work in tandem. The former is used to branch batch execution to a text string preceded by the latter, which is known as a label. A label marks the place to which `GOTO` directs batch execution.

☐ 7. The `IF` command is used for conditional branching. It can test: (1) the existence of a parameter, (2) the existence of a file, and (3) an exit code of certain DOS commands.

☐ 8. Use the popular expression `if .== %1.` to test the existence of a parameter entered on the command line. Change the number to the number of parameters expected.

- ☐ 9. Use the `FOR..IN..DO` command for looping operations. It is a compact command that can pack a number of activities in a simple line.
- ☐ 10. When a batch file executes a program, control is returned after the program exits. On the contrary, when a batch file executes another batch file, control is normally not returned. This is a chaining operation.
- ☐ 11. If you want control to return after executing another batch file, use `CALL` (or `COMMAND/C` for versions before 3.3). This becomes a nesting operation.
- ☐ 12. For a long and complex batch file, break it down to smaller modules and nest or chain them. This is easier to organize and debug. Some of the modules can be recycled, thus saving you programming time.

Review Questions

Write an answer for each question below:

1. If you don't want a line's execution displayed, what can you do? If you want to hide an entire batch file's execution, what can you do?
2. How do `REM` and `:` differ?
3. What message does the `PAUSE` command display? How can you pause a screen display and display your own message instead?
4. What is a label? How does DOS recognize a label?
5. Are labels case sensitive? If there are several labels of the same spelling, which one is recognized?
6. What is a dummy variable (replaceable parameter) and how is each dummy variable related to a parameter?
7. What does the `SHIFT` command do?
8. Explain the differences among looping, branching, nesting, and chaining.
9. What can the `IF` command test?
10. What is the difference between `CALL` and `COMMAND/C`?
11. What is the difference between a batch file calling (using `CALL`) another batch file and running another batch file?
12. What is the difference between a batch file running another batch file and running a program?
13. What are exit codes? Cite a few commands that use them.
14. Cite the commands used to:
    1. Test a file's existence
    2. Test an exit code
    3. Test a parameter's existence
15. What does the `FOR..IN..DO` command do?

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
PAUSE	Displays this: `Strike a key when ready . . .`	Displays this: `Press any key to continue . . .`
FOR..IN..DO	Versions before 3.2 can't use directory path inside ()	Can use directory path inside ()
XCOPY	Version 3.3 reports exit codes	Exit codes no longer reported
CALL	Available from Version 3.3; earlier versions must use `COMMAND/C`	Available

Using ANSI.SYS

Chapter 12

TOPICS:

ANSI Escape Sequences
PROMPT or ECHO
Controlling the Cursor
Escape Sequences and Batch Files
Setting Screen Colors and Modes
Redefining Keys
Using ANSI.BAT to Set ANSI Values
Using MEMO.BAT to Write Memos
Using MEMO1.BAT to Write Memos
Writing Phone Messages

COMMANDS COVERED:

PROMPT, ECHO

NEW TERMS AND SYMBOLS:

macro, ANSI, ANSI escape sequence, $e[, ^[[, extended keys

REQUIRED MATERIALS:

1. A practice disk
2. ANSI.SYS included in your CONFIG.SYS
3. Access to EDLIN.COM, MORE.COM, FIND.EXE
4. Ability to use EDLIN

Quick Start

✎ **To enter an ANSI command on the command line:**

```
A>prompt $e[7m
```

✎ **To enter an ANSI command from a batch file:**

```
echo ^[[7m
```

✎ **For various ways of locating the cursor:**
See Table 12.1.

✎ **To place the cursor at a particular position:**

```
A>prompt $e[row#;col#H
```

✎ **For various screen attributes:**
See Table 12.2.

✎ **To set black foreground (text) and white background:**

```
A>prompt $e[30;47m
```

✎ **To define F10 to enter `DIR` by just pressing it:**

```
A>prompt $e[0;68;"dir";13p
```

✎ **To define F12 to display `prompt $e[` (for quick use of `PROMPT`):**

```
A>prompt $e[0;134;"prompt $$e["p
```

✎ **For single-number key codes:**
See Table 12.4.

✎ **For extended key codes:**
See Table 12.5.

✎ **To quickly enter an ANSI command:**
Use the `ANSI.BAT` batch file.

✎ **To quickly write a memo:**
Use the `MEMO.BAT` or `MEMO1.BAT` batch files

✎ **To quickly write a phone message:**
Use the `MSG.BAT` batch file and accompanying key definitions.

As explained in Chapter 9, `ANSI.SYS` is a special device driver included in your DOS package. It must be installed before you can follow the chapter material. To install it, you simply include this line in the `CONFIG.SYS` file before booting:

```
device=c:\dos\ansi.sys /x
```

This line tells DOS to go to the `C:\DOS` directory to load `ANSI.SYS` stored there. If you store it somewhere else, you need to provide a different path. Do not include the `/X` switch if you are not using version 4 and an enhanced keyboard (101 keys). If you have both, the `/X` switch allows you to define the extra keys available on your keyboard.[1]

When it comes to keyboard input and screen output, the original DOS is really a Dumb Operating System. The keyboard cannot be redefined, nor does it have any **macro** capability (storing multiple keystrokes in one key). The cursor cannot be freely moved to any part of the screen. Screen colors or display modes cannot be changed.

`ANSI.SYS` has changed all that. When you have it installed, you can do the following:

1. Locate the cursor anywhere on the screen.
2. Erase a line or the whole screen.
3. Set screen display modes and colors.
4. Redefine keyboard keys.
5. Assign multiple characters to a key.

The ability to do all these things is what makes learning DOS more fun. After studying this chapter, you will no longer view DOS as the Dull Old Stuff.

This chapter explains the ANSI features in this order: ANSI escape sequences, controlling the cursor, setting graphics modes/colors, and redefining keys. After these basics are covered, we will use them in a few batch files to let you quickly write memos and take telephone messages.

ANSI Escape Sequences

- **Put `$e[` after `PROMPT` to enter an ANSI escape sequence.**
- **Put `^[[` after `ECHO` or `PROMPT` to enter an escape sequence.**

ANSI (American National Standards Institute) has adopted a set of rules to enhance input and output functions. To tap into the installed `ANSI.SYS` driver to extend keyboard and screen functions, you need to use the proper DOS commands to enter ANSI commands.

An ANSI command consists of the following components:

1. Esc (escape) character (ASCII 27)
2. Left bracket ([, or ASCII 91)
3. One or more characters or decimal numbers
4. Ending with a specific letter

[1] The `/X` switch may not affect your keyboard. On the enhanced keyboard connected to my 386 PC, I can define F11 and F12 with or without the switch.

An ANSI command starts with the Esc character, followed by a series of characters. Such a sequence of characters is often referred to as an **ANSI escape sequence**. The two terms, ANSI command and ANSI escape sequence, are thus synonymous.

To send an ANSI escape sequence to the computer, you need to use a DOS command that can understand and convert an ANSI escape sequence into action. It may be the following, which sets the screen display to reverse video:

```
A>prompt $e[7m
```

The escape character (here shown as `$e`), often referred to as Esc or ESC, is not the same as your typing these three letters or pressing the Esc key on the keyboard. DOS will understand it only if you send the ASCII 27 (1B hex) value.

How do you tell DOS to go by ASCII 27? There are two ways, depending on what DOS command you use. These are shown below:

1. `$e` Used with `PROMPT` only; entered from command line, EDLIN, or `COPY CON`
2. `^[` Used with `PROMPT`, `ECHO`, `COPY`, `MORE`, `TYPE`; entered in EDLIN only

When used with `PROMPT`, DOS interprets $e as ASCII 27. When this string is followed by `[`, DOS expects an ANSI parameter. Other commands, however, treat `$e[` as pure and simple text without any special meaning.

You can enter `$e` on the command line, `COPY CON` screen, or EDLIN; `^[`, however, can be entered only in EDLIN or another word processor, not on the command line or `COPY CON` screen. (If you use WordPerfect, hold down Ctrl and press V and [; the screen will then display `^[`.)

To enter an escape character in EDLIN, you need to press Ctrl-V to insert `^V` and then the left bracket ([); the screen then shows `^V[`. When you use the `L` command to list text lines, you see `^[`. (See Chapter 10 for more details.) After ^[(ASCII 27), you need to add an extra [, as required by ANSI. With an escape character and a left bracket, you can then enter parameters to form a meaningful ANSI escape sequence.

If `ANSI.SYS` is not installed, these escape sequences are displayed as text or special characters and are not converted to any action. These commands are meaningless to plain old DOS.

PROMPT or ECHO

✎ **In a batch file, `ECHO` is better than `PROMPT` for entering ANSI escape sequences.**

The most common ways to enter ANSI escape sequences are through `PROMPT` and `ECHO`. `PROMPT` is used most commonly on the command line, and `ECHO` in batch files.

On the command line, you have no choice but to use `PROMPT`. From a batch file, however, you can choose to use `PROMPT` or `ECHO`. These two behave quite differently. `PROMPT` will alter the DOS prompt. When `ECHO` is off, `PROMPT` will work haphazardly. `ECHO`, on the other hand, will not change the DOS prompt and will work well whether `ECHO` is on or off.

Following are a few examples of the differences between PROMPT and ECHO in entering ANSI commands. All the examples are intended to assign dir to the F8 key, type to F9, and copy to F10. Redefining (reassigning or remapping) keys will be explained later. For now, we use these examples to illustrate ECHO and PROMPT.

PROMPT and $e[

The first example uses PROMPT with $e[:

```
1: @echo off
2: prompt $e[0;66;"dir "p
3: prompt $e[0;67;"type "p
4: prompt $e[0;68;"copy "p
```

Lines 2 and 3 will fail, and only line 4 will result in redefinition; the previously assigned values of F8 (line 2) and F9 (line 3), if any, will remain. If line 1 is deleted, namely ECHO set on, all redefinitions will be successful. Line 4 (the last of the list) will also be in the environment regardless of the ECHO status. If you enter the SET command, a list, including the following, will appear:

```
PROMPT=$e[0;68;"copy "p
```

The previous PROMPT line in the environment, if any, is replaced by this line. The familiar A> prompt also disappears. If you enter PROMPT alone, the default A> prompt reappears, and the environment PROMPT disappears.

PROMPT and ^[[

In order to see how PROMPT's behavior is affected, let us modify the above batch file by replacing each $e[with ^[. In the batch file below, you are to press Ctrl-V to display ^V, and then follow it with [. When you use the L command, the escape character becomes ^[, as shown below:

```
1: @echo off
2: prompt ^[[0;66;"dir "p
3: prompt ^[[0;67;"type "p
4: prompt ^[[0;68;"copy "p
```

This batch file will work (or fail) the same as the earlier one. The only difference is that the environment shows this line:

```
PROMPT=
```

The escape sequence after PROMPT is not displayed.

If you use TYPE to show the batch file, the escape character (^[) and all the data after them are also invisible, unlike $e[, which are visible like any text string.

ECHO and ^[[

Let us now replace every PROMPT with ECHO:

```
1: @echo off
2: echo ^[[0;66;"dir "p
3: echo ^[[0;67;"type "p
4: echo ^[[0;68;"copy "p
```

This batch file, unlike those before, will work well. All the key definitions will succeed. The DOS prompt will not change. And the environment's PROMPT, if any, will not be affected.

The above examples show that using ECHO is preferable over PROMPT when in a batch file. Since ECHO can be turned off, the screen remains uncluttered when keys are being redefined.

Note: If you need to use metastrings, you must use PROMPT. ECHO does not recognize them. This will be demonstrated shortly.

From now on, we will use ECHO in a batch file when an escape sequence is to be sent. On the command line, however, we will continue to use PROMPT in combination with $e[, because the escape character recognized by ECHO (^[) cannot be entered from the command line.

Tips & Tricks

Saving the Prompt

Whenever the PROMPT command is entered, either on the command line or from a batch file, the DOS prompt disappears. This could be disorienting to novices. To prevent this, you can add necessary metastrings to automatically bring back the prompt, such as:

```
A>prompt $e[7m$p$g
A:\>
```

After setting the reverse video, the PG strings display the current directory of the default drive, plus the > sign.

In the future, you can also add these or similar strings after setting cursor location, changing screen colors, or defining keys, such as:

```
A>prompt $e[0;68;"dir";13p$n$g
A>
```

This command defines F10 as dir plus a carriage return. After the definition, the A> prompt is restored. Without the two metastrings at the end, the prompt would disappear.

Drill

_____ 1. DOS will accept and implement an ANSI escape sequence with or without ANSI.SYS installed. True or false?

_____ 2. To enter Esc in an ANSI escape sequence, you can press the Esc key on the keyboard. True or false?

_____ 3. The command `A>echo ^V[[7m`, entered on the command line, will change your screen display to reverse video. True or false?

_____ 4. If you enter the command `A>prompt $e[7m`, the DOS prompt will disappear. True or false?

_____ 5. The command in Question 4 will cause the line `PROMPT=$e[7m` to be included in the DOS environment. True or false?

_____ 5. Both `ECHO` and `PROMPT` will work with metastrings. True or false?

_____ 6. In the batch file below, both `PROMPT` commands will be successfully implemented. True or false?

```
@echo off
prompt $e[0;68;"dir"p
prompt $e[7m
```

Controlling the Cursor

✎ **Use a cursor-control escape sequence to position the cursor on the screen or erase a screen display.**

The ANSI escape sequences for controlling the cursor are listed in Table 12.1. You can send an escape sequence appearing on the left to achieve a special effect described on the right.

Table 12.1 ANSI Escape Sequences—Cursor and Erasure

Cursor Control	
Esc [#;#H	Cursor row-col position; default: 1,1
Esc [#;#f	Same as Esc [#,#H
Esc [#A	Cursor up by # lines; default 1
Esc [#B	Cursor down by # lines; default 1
Esc [#C	Cursor forward (right) by # columns; default 1
Esc [#D	Cursor backward (left) by # columns; default 1
Esc [6n	Device status report
Esc [s	Save cursor position
Esc [u	Restore cursor position
Display Erasure	
Esc [2J	Erase display; clear screen
Esc [K	Erase from cursor to end of line

The # character in the table should be replaced with a decimal number representing a row or column position on the screen. Where there are two such characters, the first is for row number and the second for column number.

Warning: Beware of case. Each letter must be entered as shown in the table. If you use a different case, DOS will not understand your instruction and thus will fail to implement your wish.

Let us try some concrete examples. Follow these steps:

1. Move the cursor to the middle of the screen:

```
A>prompt $e[12;40H
```

The cursor now goes to the center of a 25x80 screen; the familiar DOS prompt, as usual, disappears. This new location will not interfere with your giving instructions to DOS.

2. Display directory entries:

```
A>dir
```

The screen should be full, but the cursor remains in the middle of the screen instead of at the usual bottom of the screen.

3. Clear the screen:

```
A>cls
```

The screen is cleared, but the cursor remains in the same place.

4. Display the contents in the environment:

```
A>set
```

In addition to whatever is previously stored in the DOS environment, you will find this line:

```
PROMPT=$e[12;40H
```

This is what causes the cursor to stay in the same place. Unless this value is changed, the cursor stays where it is no matter what you do.

5. Clear the environment `PROMPT`:

```
A>prompt
```

Entering the `PROMPT` command alone or in combination with parameters will cause the `PROMPT` line in the environment to disappear or change its value. The cursor position will be controlled by either nothing or a new value. After the above command, the cursor goes back to the left margin and begins to behave normally.

6. Clear the screen:

```
A>prompt $e[2J
```

The screen is cleared and the cursor goes to the top left corner of the screen. The line below also enters the environment:

```
PROMPT=$e[2J
```

7. Display directory entries:

```
A>dir
```

The display is immediately erased. As long as this PROMPT value remains in the environment, the screen will continue to behave this way.

8. Clear the environment PROMPT:

```
A>prompt
```

Instead of erasing the whole screen, you may want to erase the current line from the cursor location to the end. This command will do just that:

```
A>prompt $e[k
```

Most of the escape sequences shown in Table 12.1 have default values. If you do not enter a number in place of #, DOS assumes that you want 1. For example, if you want to park the cursor at the top left corner of the screen, you can simply enter this command:

```
A>prompt $e[H
```

The missing row and column values default to 1. The same thing is true when you want to move the cursor up, down, left, or right. If no value is entered, 1 is presumed.

Sometimes it is useful to save a cursor position and then return to it. When `Esc [u` is encountered, the cursor returns to where it was after `Esc [s` was most recently executed. This technique will be used in a batch file in the next section.

Escape Sequences and Batch Files

✎ **On many occasions, it is preferable to enter ANSI escape sequences by using a batch file rather than by typing on the command line.**

Entering ANSI escape sequences is quite laborious, requiring numerous keystrokes. To save time, you can enter the necessary keystrokes in a batch file. When the batch file is run, all the stored keystrokes will be quickly replayed.

Combining Metastrings and ANSI Commands

In Chapter 5, you learned to use the PROMPT command and metastrings to do a variety of tricks with the DOS prompt. We can now combine that knowledge with what has been discussed so far in this chapter to do something fancy.

Use EDLIN to create a file named SHOWTIME.BAT. Then enter the following lines:

```
1: @echo off
2: cls
3: prompt $e[s$e[7m$e[1;67H$d$e[2;67H$t$e[0m$e[u$n$g
```

Double check your typing. When everything is correct as shown, exit EDLIN. Run the batch file:

```
A>showtime
```

The upper right corner of the screen displays in reverse video something like the following:

```
Mon 06-26-1990
19:42:08.74
```

The cursor also stays in the top left corner, after the default DOS prompt.

The items in line 3 of the batch file are separately explained below:

```
$e[s          Save cursor position at execution time
$e[7m         Turn on reverse video
$e[1;67H      Position cursor in row 1, column 67
$d            Display the current date
$e[2;67H      Position cursor in row 2, column 67
$t            Display the current time
$e[0m         Turn off all colors; restore white on black
$e[u          Restore cursor position saved by $e[s
$n            Display default drive letter
$g            Display greater than sign (>)
```

Note: Notice that you can combine lots of escape sequences and mix them with metastrings. Each ANSI parameter has to be placed after `$e[`, and each metastring preceded by `$`. The `PROMPT` in the above batch file cannot be replaced with `ECHO`, which cannot manipulate metastrings such as `$D$T`.

There are numerous possible ways you can create fancy prompts by combining metastrings and ANSI commands. Here is another example:

```
1. @echo off
2: cls
3: prompt $e[44;5m * * $e[41m▬▬$_$e[44;5m * * $e[41m▬▬$e[K$e[m$p$g
```

Enter line 3 as a single line (do not break it up). The solid bar is made up with ASCII 223 characters (hold down Alt and press 223 on the numeric pad. When you run this batch file (name it `FLAG.BAT`), a blinking colorful flag will become your prompt style. If you do not understand the various numbers, consult the metastring table in Chapter 5 and various tables in this chapter.

Displaying a Message

Another trick you could use in your batch file is to build a routine to clear the screen and display whatever message you intend for yourself or another user. The following fragment could be put in your longer batch file:

```
1: ^[[2J
2: ^[[12;34HWarning!
3: ^[[14;26HYour message is put here.
```

If you write this as a separate file and name it `MESSAGE`, you can use the following command to display the message:

```
A>type message
```

The screen will be cleared and the text displayed in the middle. If another batch file calls for displaying this message, you could use the following method:

```
1: @echo off
2: type message
3: echo ^[[24;1f
4: pause
5: rem . . . .
```

This batch file will clear the screen and display the message. After that, the `PAUSE` message will be displayed at the bottom of the screen, namely line 24, column 1. The `f` in line 3 can be changed to `H` if you wish; they are interchangeable.

Practice 12-1

- ☐ 1. Use the `PROMPT` command to position the cursor at row 1, column 1 and at the same time clear the line after the default DOS prompt.
- ☐ 2. Use the `PROMPT` command to position the cursor at the bottom left corner of the screen. The default prompt must also be displayed.
- ☐ 3. Use the `PROMPT` command to clear the screen and at the same time position the cursor at the top left corner of the screen. Then restore the default prompt and the regular screen behavior.
- ☐ 4. Use the `PROMPT` command to display the current time in reverse video in row 1, column 35. Then restore the regular DOS prompt.
- ☐ 5. Use EDLIN to create a batch file that will display in the middle of a clear screen the message `Hello, World.` No prompt is to be displayed. The cursor should be located at the bottom left corner.

Drill

Which of the following letters should replace ? in questions 8–11:
a. A, b. B, c. C, d. D.

_____ 8. Cursor right—`Esc [#?`

_____ 9. Cursor left—`Esc [#?`

_____ 10. Cursor up—`Esc [#?`

_____ 11. Cursor down—`Esc [#?`

_____ 12. The command `A>prompt $e[H` will:
a. cause the DOS prompt to disappear
b. move the cursor to the top left corner
c. do both 1 and 2
d. do neither 1 or 2

_____ 13. The command `A>prompt $e[2J$d/$t` will:
a. clear the DOS prompt
b. clear screen and display date/time
c. do both 1 and 2
d. do nothing

_____ 14. What happens if you have a file named `X` which contains `^[[7m` and you enter the `TYPE X` command?
a. The computer beeps.
b. An error message is shown.
c. The screen changes to reverse video.
d. Nothing happens.

_____ 15. After the following two commands, the current directory's entries will stay on the screen. True or false?

```
A>prompt $e[2J
dir
```

Setting Screen Colors and Modes

✎ **Set colors or display modes by specifying attributes and ending with `m` (for monitor).**

Table 12.2 ANSI Escape Sequences—Graphics

Set Graphics Rendition (SGR)

`Esc [#;...;#m` Set graphics attributes as follows:

Parameter	Effect
0	Normal; black on white
1	Bold; high intensity
4	Underline; mono only
5	Blink
7	Reverse video
8	Concealed; invisible
30	Black foreground
31	Red foreground
32	Green foreground
33	Yellow foreground
34	Blue foreground
35	Magenta foreground
36	Cyan foreground
37	White foreground
40	Black background
41	Red background
42	Green background
43	Yellow background
44	Blue background
45	Magenta background
46	Cyan background
47	White background

Set Mode (SM)

`Esc [=#h` Set screen mode with specific parameter

Parameter	Effect
0	40x25 black and white
1	40x25 color
2	80x25 black and white
3	80x25 color
4	320x200 color
5	320x200 black and white
6	640x200 black and white
7	Wrap at end of line

Reset Mode (RM)

`Esc [=#l` Same as SM, except 7, which turns off wrap; in a long line, the screen shows only one line

ANSI.SYS allows you to set your monitor with a variety of attributes, most notably colors and the number of lines and characters displayed. Table 12.2 shows the formats and attributes. (Your monitor may not be capable of accepting the available attributes.)

As with cursor control, explained in the previous section, you need to send escape sequences to your computer. Here is an example:

```
A>prompt $e[30;47m
```

This command sets the text color to black and the background color to white, achieving the same effect as reverse video on both color and monochrome monitors. Do not forget the m at the end; it has to be lowercase.

Tips & Tricks

ECHO On/Off

When ECHO is off on the command line (showing no prompt), using PROMPT to enter an ANSI command will not work immediately. Consider the following sequence of commands:

```
A>echo off
prompt $e[7m
```

This will send the second line to the environment, but not affect the screen color right away. If ECHO is now turned on, the screen will be changed to reverse video. If ECHO is then turned off, the reverse video remains.

This observation also applies to metastrings such as $D and to other ANSI actions such as redefining a key.

You can turn ECHO on/off at will to enable/disable a PROMPT command in the environment intended to control the cursor. When ECHO is off, a cursor-control PROMPT, such as PROMPT $e[2J, has no effect and the cursor behaves normally.

Instead of using PROMPT with $e[to set screen colors, you can use ^[[with ECHO, TYPE, MORE, or COPY CON, such as in this batch file:

```
1: @echo off
2: ^[[30;47m
```

Suppose you name this file COLOR.BAT. You can enter this command to set colors:

```
A>type color.bat
```

The TYPE command can also be replaced with MORE or COPY CON, such as:

```
A>more < color.bat
A>copy color.bat con
```

If you run this batch file, such as:

```
A>color
```

you will get an error message. DOS does not consider the second line to be a valid command. The following change will rectify the problem:

```
1: @echo off
2: echo ^[[30;47m
```

If you need to change the screen mode, you can use a command like this:

```
A>prompt $e[=1h
```

This command sets the screen to 40x25 color. The `h` can be replaced by `l` with the same effect. To get back to 80x25 color, enter this:

```
A>prompt $e[=3h
```

Normally, when you enter a long line, DOS automatically goes down to the beginning of a new line for you to continue. If for some reason you do not want to wrap, try this:

```
A>prompt $e[=7l
```

After that, the cursor continues to stay in the same line at the right margin. DOS continues to accept your typing, but each new character replaces the old one displayed. To restore the default, this command will do:

```
A>prompt $e[=7h
```

Armed with this knowledge, you can now spruce up the message display explained in the previous section (name it `MESSAGE.BAT`) and make it flashy:

```
 1:*@echo off
 2: echo ^[[2J
 3: echo ^[[=1h
 4: echo ^[[5;31;43m
 5: echo ^G^G^G
 6: echo ^[[11;17HWarning!
 7: echo ^[[13;8HYour message is put here.
 8: echo ^[[22;1H
 9: echo ^[[0m
10: pause
11: echo ^[[=3h
```

When you run this batch file, the computer beeps and the message in lines 6–7 will appear in large size and blinking in bright colors. The `PAUSE` message appears at the bottom in white on black.

You can understand most of the lines by referring to Table 12.2. Line 2 clears the screen. Line 3 sets 40x25 mode and line 11 sets it back to 80x25 when it is over. Line 4 sets blinking red foreground and yellow background. Line 5 causes a beep three times. Line 8 locates the cursor near the bottom of the screen. Line 9 turns off all color attributes.

Practice 12-2

☐ 1. Modify question 5 in Practice 12-1 by displaying green text on yellow background; restore the default value at the end. If you have a monochrome monitor, make the text blink and underlined.

☐ 2. Modify the above to display the word HELLO vertically and nothing else on a black-on-white screen.

Drill

_____ 16. The command `A>prompt $e[5` will:
- a. cause text to blink
- b. turn off the prompt
- c. generate an error message
- d. do nothing

_____ 17. The command `A>prompt $e[7;m` will:
- a. turn on reverse video
- b. turn on high intensity
- c. turn off all attributes
- d. do nothing

_____ 18. The command `A>prompt $e[1;5;7m` will produce:
- a. high intensity, blink, reverse video
- b. reverse video only
- c. high intensity only
- d. none of the above

_____ 19. The command `A>prompt $e[m;7m` will produce:
- a. reverse video
- b. blink
- c. default attribute
- d. an error message

Redefining Keys

✎ **Most of the keyboard keys can be redefined (reassigned).**

✎ **Redefine a key by entering an escape sequence containing the first item as the original key, the second item as the new key, and ending with p.**

With `ANSI.SYS` installed, you can redefine almost all the keys you see on your keyboard and their many combinations. Since many keys or key combinations are either unused by DOS or rarely used, you can redefine them to speed up much of your work.[2]

Table 12.3 ANSI Escape Sequences—Key Definition[2]

`Esc [#;#p`	Assign value (2nd #) to key (1st #)
`Esc [#;"string"p`	Assign a string to a key
`Esc [#;"dir";13p`	Assign `dir` and Enter to a key
`Esc [0q`	To disable defining F11 & F12
`Esc [1q`	To enable defining F11 & F12

Table 12.3 shows a number of ways you can redefine keys. If you want to redefine a key appearing on the keyboard, just put the original and its new definition in quotes, such as:

```
A>prompt $e["[";"{"p
```

After this, pressing the [key always shows {. To restore its original definition, try this:

```
A>prompt $e[91;91p
```

This will restore [to its original ASCII value of 91.

Table 12.4 Single-Number Key Codes

1–26	Ctrl-A–Ctrl-Z
8	Backspace, Shift-Backspace
9	Tab
10	Ctrl-Enter
13	Enter
27	Esc

[2] The Q parameter may not affect your keyboard. On the enhanced keyboard connected to my 386 PC, the OQ parameter does not prevent me from defining F11 and F12.

Some keys, such as Esc or ^C, have no literal existence like the visible keys appearing on the keyboard; these are shown in Table 12.4. To redefine such a key, you need to enter its ASCII value, such as:

```
A>prompt $e[27;"Escape"p
```

This assigns the `Escape` string to the Esc key. If you press Esc now, `Escape` appears. To restore its original definition, try this:

```
A>prompt $e[27;27p
```

After this command, pressing Esc shows the usual \ sign.

Warning: Do not assign another value to the Enter key (ASCII 13, which is also the value of ^M), such as this example:

```
A>prompt $e[13;"dir"p
```

After this command, you can no longer use the Enter key for its original purpose. Every time you press Enter, `dir` will show up. Unless you have assigned ASCII 13 to another key, you can no longer enter ASCII 13 to register a DOS command. The only way out is to reboot.

There are also **extended keys**, which require you to enter two numbers to redefine them. The first number is always 0 and the second is based on Table 12.5. Here is an example:

```
A>prompt $e[0;68;"prompt $$e["p
```

This command redefines F10 as `prompt $$e[`. The two dollar signs are to display the second dollar sign; without two, none will be displayed. After this command, pressing F10 will show this:

```
A>prompt $e[
```

You can now enter numbers and characters to complete an escape sequence. This will save you a lot of time as you experiment with ANSI commands.

You can assign multiple values to a key, such as:

```
A>prompt $e[0;93;"dir";13p
```

This assigns to Shift-F10 `dir` plus a carriage return (ASCII 13 has the same effect as pressing Enter from the keyboard). If you now press Shift-F10 (without pressing Enter), the current directory's entries will be immediately displayed.

If you have an enhanced keyboard, you can take advantage of F11 and F12 and their various combinations. Here is an example:

```
A>prompt $e[0;134;"cls$_"p
```

After this command, pressing F12 will immediately clear the screen.[3] The $_ metastring, as explained in Chapter 5, has the effect of pressing the Enter key. You can thus use this metastring or ASCII 13 as used in the previous command.

[3] If your keyboard does not respond as defined, you may need a short TSR program listed in *PC Computing* (June 1990, p. 242).

Table 12.5 Extended Key Codes

Second Code	Key
1	Alt-Esc
3	Null character, ASCII 0
14-15	Alt-Backspace, Shift-Tab
16-25	Alt-Q, W, E, R, T, Y, U, I, O, P
26-28	Alt-[,], Enter
30-41	Alt-A, S, D, F, G, H, J, K, L, ; ', ‘
43-50	Alt-\, Z, X, C, V, B, N, M, ,, ., /
55, 57	Alt-Key *, space bar
59-68	F1–F10
71-83	Home, Up, PgUp, Alt-Key -, Left, Key 5, Right, Alt-Key +, End, Down, PgDn, Ins, Del
84-93	Shift-F1–Shift-F10
94-103	Ctrl-F1–Ctrl-F10
104-113	Alt-F1–Alt-F10
114-119	Ctrl-PrtSc, Ctrl-Left, Ctrl-Right, Ctrl-Right, Ctrl-PgDn, Ctrl-Home
120-132	Alt-1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, =, Ctrl-PgUp
133-140	F11, F12, Shift-F11, Shift-F12, Ctrl-F11, Ctrl-F12, Alt-F11, Alt-F12
141-147	Ctrl-Up, Ctrl-Key -, Ctrl-Key 5, Ctrl-Key +, Ctrl-Down, Ctrl-Ins, Ctrl-Del
148-150	Ctrl-Tab, Ctrl-Key /, Ctrl-Key *
151-153	Alt-Home, Alt-Up, Alt-PgUp
155,157	Alt-Left, Alt-Right
159-163	Alt-End, Alt-Down, Alt-PgDn, Alt-Ins, Alt-Del
164-166	Alt-Key /, Alt-Tab, Alt-Key Enter

(Key = numeric keypad. Some key combinations apply only to an enhanced keyboard.)

Practice 12-3

- ☐ 1. Redefine the Esc key on the keyboard so that when you press it alone the current directory's entries will immediately appear.
- ☐ 2. Restore the Esc key to its original state.
- ☐ 3. Redefine the Home key so that by pressing it alone you can display the root directory's entries.

☐ 4. Redefine the Alt-Esc key combination so that when you are editing a line in EDLIN you can press this key combination to break out of the line and exit EDLIN at the same time.

Drill

_____ 20. The command `A>prompt $e[0;1;3p` will define:
   a. `^A` as `^C`
   b. `^A` as 3
   c. Alt-Esc as `^C`
   d. nothing

_____ 21. The command `A>prompt $e[26;8p` will define:
   a. `^Z` as Backspace
   b. `^A` as `^Z`
   c. Alt-P as `^Z`
   d. nothing

_____ 22. The command `echo [["b";"B"p`, entered from a batch file, will redefine `b` as `B`. True or false?

_____ 23. The following will restore the above key to its original definition. True or false?

```
echo ^[[98;98p
```

_____ 24. The command `A>prompt $e[0;16;"dir"p$e[0;17;"type"p` will define:
   a. Alt-Q
   b. Alt-W
   c. both 1 and 2
   d. neither 1 or 2

Using `ANSI.BAT` to Set ANSI Values

✎ **Use a batch file to speed up setting screen colors and/or redefining keys.**

Using the `PROMPT` command on the command line to set screen colors/modes or to redefine keyboard keys, as shown in previous sections, requires a great deal of repeated typing. Furthermore, the DOS prompt disappears every time you enter the `PROMPT` command. These problems can be remedied with a batch file like this:

```
1: @echo off
2: rem ANSI.BAT--set ANSI color and key definition
```

```
 3: if .==.%1 goto HELP
 4: echo ^[[%1;%2;%3;%4;%5
 5: goto END
 6: :HELP
 7: echo/
 8: echo You forgot to enter parameters, such as:
 9: echo       %0 7m (for reverse video)
10: echo       %0 30;47m (black text, white background)
11: echo       %0 0;68;"dir"p (define F10 as "dir")
12: echo/
13: :END
```

If you just enter the batch file name alone, namely ANSI, line 3 becomes true and line 6 is executed. The error message in lines 8–11 is displayed. Lines 7 and 12 create empty lines to make the error message more readable.

If a parameter is entered after ANSI, line 3 is ignored and line 4 is executed. Each dummy variable is replaced by a parameter entered on the command line. For example, you could enter this command:

```
A>ansi 0;68;"dir/p";13p
```

In this command, four parameters separated by semicolons follow the batch file name. DOS responds by executing line 4 this way:

```
echo ^[[0;68;"dir/p";13p;
```

This causes F10 to be defined with `dir/p` plus a carriage return.

You can, if you wish, use a space instead of semicolon as a parameter separator, such as:

```
A>ansi 0 68 "dir|sort" 13p
```

This will assign `dir|sort` plus a carriage return to F10. Pressing F10 after this command will display sorted directory entries.

The arrangement in line 4 allows five parameters to be entered on the command line. If fewer are entered, the screen displays the extra semicolons and no harm is done. If more are entered, your command may not be executed. If you need to enter more parameters, you can increase the number of dummy variables up to nine.

Since the batch file does not use the `PROMPT` command, running it will not cause the DOS prompt to disappear or change the value of `PROMPT`, if any, in the environment.

Note: This batch file can also be used to set screen modes/colors or control cursor positions; just make sure to end each command with the proper letter. It is a simple and versatile ANSI companion; don't use ANSI without it.

Practice 12-4

- ☐ 1. Use `ANSI.BAT` to redefine the PgUp key so that when you press it the parent directory's entries are displayed.
- ☐ 2. Use `ANSI.BAT` to redefine `X` as `Y` and then restore the original.

- ☐ 3. Use `ANSI.BAT` to redefine Ctrl-F10 as `dir|sort|more`, plus a carriage return.
- ☐ 4. Restore Ctrl-F10 to its original value.

Tips & Tricks

The Ghost of `PROMPT`

`If you use ECHO` to redefine a key (or change screen colors), it will not negate an identical command entered with `PROMPT`. Suppose you have most recently entered this command to redefine F10:

```
A>prompt $e[0;68;"dir"p
```

If you try to redefine F10 with a batch file command like below:

```
echo ^[[0;68;"type "p
```

the key will not be redefined. The reason is that this line still stays in the environment:

```
PROMPT=$e[0;68;"dir"p
```

If you enter `PROMPT` alone or in combination with other parameters, the above line will either disappear or change its value. If you then redefine F10 (or any other key) with `ECHO`, there will be no obstacle to your move.

Using `MEMO.BAT` to Write Memos

✎ **Use a batch file for quick writing of memos.**

If you know how to redefine keys, you can take advantage of the knowledge to simplify and speed up your work. For example, if you often write short memos while working with DOS, you can use a batch file like the one below to quickly do the work:

```
1: @echo off
2: rem MEMO.BAT--using COPY CON to write memos
3: if .==.%1 goto HELP
4: echo ^[[0;50;"                    MEMO";13;13;"To: "p
5: echo ^[[0;33;"From: "p
6: echo ^[[0;32;"Date: "p
7: echo ^[[0;31;"Subject: "p
8: cls
9: echo ^[[7m
```

```
10: echo Alt-M to start memo; Alt-F = From, Alt-D = Date, Alt-S
    = Subject
11: echo The file being created is: %1. Press F6 and Enter to
    save file.
12: if exist %1 echo %1 already exists. It'll be replaced if
    you continue.
13: if exist %1 echo Press ^C if you wish to abort.
14: echo ^[[0m
15: copy con %1
16: goto END
17: :HELP
18: echo/
19: echo You must enter a file name after MEMO. Try again.
20: echo/
21: :END
```

You must run this batch file with a file name, such as:

```
A>memo carl
```

The file named CARL will be created. If no file name is entered, this message appears and the batch file ends:

```
A>memo

You must enter a file name after MEMO. Try again.

A>_
```

After the batch file starts, a two-line message appears:

```
Alt-M to start memo; Alt-F = From, Alt-D = Date, Alt-S = Subject
The file being created is: carl. Press F6 and Enter to save file.
```

The message also appears in reverse video, which is turned on by line 9 and off by line 14.

If the file name you enter matches one in the current directory, you will be warned of that fact. If you proceed, the new one will replace the old. If you do not want that to happen, press Ctrl-C to abort the batch file and start again with a new name.

You can now utilize any of the four keys already defined by the batch file to speed up your work. If you press Alt-M, the first two lines shown below appear and the cursor is positioned for you to enter a name.

```
MEMO

To: Carl
From: Tim
Date: 6/22
Subject: Sales
```

After you type the first line and press Enter, you can press Alt-F, Alt-D, and then Alt-S and enter each line. After that, you can type your message below. Press F6 and Enter to save the memo.

Lines 4–7 redefine the four keys to be used in writing a memo. After the memo is finished, these definitions remain. If you wish, you can use them for whatever purpose. If you want to restore their original definitions, add a simple routine between lines 15 and 16.

Practice 12-5

- ☐ 1. Modify `MEMO.BAT` by replacing all the Alt keys with Ctrl. (Use `^N` for `Alt-M`. Do not redefine `^M`, which is the same as Enter. If you did, you would no longer be able to enter another command.)
- ☐ 2. Use the modified batch file to write a memo.
- ☐ 3. Print the memo by using `COPY`.

Using `MEMO1.BAT` to Write Memos

✎ **Use a batch file to load EDLIN for quick writing of memos.**

If you feel `COPY CON` is too restricting in typing text, you can modify the batch file to let you use EDLIN. Our modified batch file, named `MEMO1.BAT`, is shown below:

```
 1: @echo off
 2: rem MEMO1.BAT--using EDLIN to write memos
 3: rem EDLIN must be in the current dir
 4: if not exist edlin.com echo EDLIN not found in the
    current directory.
 5: if not exist edlin.com goto END
 6: if .==.%1 goto HELP
 7: cls
 8: echo ^[[7m
 9: echo Alt-M to start memo; Alt-F = From, Alt-D = Date,
    Alt-S = Subject
10: echo The file being created is: %1
11: if exist %1 echo %1 already exists. It'll be replaced
    if you continue.
12: if exist %1 echo Press a key to continue or ^C to
    abort.
13: if exist %1 pause > nul
14: echo ^[[0m
15: echo ^[[0;50;"                    MEMO";13;13;"To: "p
16: echo ^[[0;33;"From: "p
17: echo ^[[0;32;"Date: "p
18: echo ^[[0;31;"Subject: "p
19: edlin %1
20: echo/
21: goto END
22: :HELP
```

```
23: echo/
24: echo You must enter a file name after MEMO1. Try again.
25: echo/
26: :END
```

Lines 4–5 handle the situation when EDLIN.COM is not found in the current directory. A message is displayed and the batch file ends. If EDLIN is found, a message similar to MEMO.BAT appears, showing the redefined keys available for use. If no matching file exists, EDLIN is started immediately and the screen displays the New file message. You can enter I to start the first line. Pressing Alt-M will start the beginning lines of a memo. Other steps are the same as in MEMO.BAT.

If an old matching file exists, this message appears:

```
Alt-M to start memo; Alt-F = From, Alt-D = Date, Alt-S = Subject
The file being created is: carl
carl already exists. It'll be replaced if you continue.
Press a key to continue or ^C to abort.
```

If you press ^C, EDLIN is not started and the batch file is ended. Pressing any other key will start EDLIN and display the End of input file message. You can then enter the L command to show the file and make necessary changes.

MEMO1.BAT has one advantage over MEMO.BAT. It allows you to edit lines after pressing Enter. Use it if your memos are fairly elaborate and require editing. For short memos, MEMO.BAT is quick and easy.

Writing Phone Messages

✎ **Use a series of batch files to quickly take, save, and print phone messages.**

Imagine that you are working with DOS, a database, and/or a spreadsheet program. Suddenly, the phone rings and you need to take a message. If you are in DOS, you can use the batch files below to quickly complete the task. If you are somewhere else, go to DOS without exiting your program and use the batch files.

To do that, you need to create three batch files. It sounds complicated. It may be a little elaborate. But once you have the system set up, everything becomes automated and you can save a lot of time.

Basically, we intend to set up a system in which you can press F10 to immediately go to the COPY CON screen where you can start typing right away. After you exit, you can, if you wish, press Shift-F10 to print the message just created; or you can print several related files at a later time.

The first batch file is a simple one:

```
1: @echo off
2: rem F10.BAT--define F10 and Shift-F10
3: echo ^[[0;68;"msg m0 m1 m2 m3 m4 m5 m6 m7 m8 m9";13p
4: echo ^[[0;93;"printit";13p
```

It redefines two keys, F10 and Shift-F10. Shift-F10 will run a batch file as explained below. F10 will enter the following command:

```
A>msg m0 m1 m2 m3 m4 m5 m6 m7 m8 m9
```

It runs the batch file named MSG.BAT, which is explained below, plus 10 parameters. These parameters are used as file names. In the beginning, the first file created (message taken) will be saved as M0. After that, the next file will be named M1, and so on. This arrangement allows 10 separate messages to coexist at any time.

The MSG.BAT file resembles in part MEMO.BAT explained earlier. Two keys are defined. The Date item is eliminated; date and time data will be automatically added by the batch file.

Line 5 provides a safeguard. When all the 10 parameters are used up, you can no longer create files. When all the files named M0-M9 are detected, %1 is assigned no value and COPY CON cannot create a file. At this point, batch execution branches to the HELP messages and ends.

```
 1: echo off
 2: rem MSG.BAT--using COPY CON to take messages
 3: :AGAIN
 4: if exist %1 goto NEXT
 5: if .==%1. goto HELP
 6: echo ^[[0;50;"                              MESSAGE";13;13;"To: "p
 7: echo ^[[0;33;"From: "p
 8: cls
 9: echo ^[[7m
10: echo Alt-M to start message; Alt-F = From
11: echo The file being created is: %1
12: echo Press F6 and Enter to end, or ^C to abort.
13: echo ^[[m
14: copy con %1
15: echo ¦ more ¦ date ¦ find "C" > %1
16: echo ¦ more ¦ time ¦ find "C" > %1
17: goto END
18: :NEXT
19: shift
20: goto AGAIN
21: :HELP
22: echo/
23: echo You have run out of available parameters
24: echo You must erase some of M0...M9 files.
25: echo You can press Shift-F10 to print and erase them.
26: echo/
27: :END
```

Line 4 is a little tricky and requires explanation. It tests to see whether a particular file exists in the current directory. If M0, the first parameter entered on the command line, does not exist, then line 4 is ignored and M0 is used in line 14 as the name of the file to be created by COPY CON.

When you run the batch file another time, line 4 will detect the existence of M0 and branch to line 18. Line 19 then shifts to the next parameter, namely M1, thus assigning M1 to %1. Line 4 again tests M1's existence. Since it does not already exist, it becomes the next file to be created.

After you exit the COPY CON screen, lines 15–16 add the date and time data to the end of the file just created. This saves you the trouble of having to add them when you are busily taking a message.

Make sure MORE.COM and FIND.EXE are stored in the current directory or accessible through the PATH command. MORE will have the effect of pressing the Enter key after DATE/TIME is entered, thus obviating the need to press this key manually. FIND will look for the string starting with C. Finally, ECHO will append the found string to the named file.

When you press F10 to start, the screen displays in reverse video the top three lines shown below. You can then press Alt-M to begin your work. After entering the first line, press Alt-F to begin the second line. When you finish taking the message, the screen resembles that below:

```
Alt-M to start message; Alt-F = From
The file being created is: m2
Press F6 and Enter to end, or ^C to abort.

                    MESSAGE

To: George
From: Garage

Your car has been fixed. Call 909-AUTO.

^Z
        1 File(s) copied
```

If you use the TYPE command to display the file just created, it shows something like this:

```
                    MESSAGE

To: George
From: Garage

Your car has been fixed. Call 909-AUTO.

Current date is Thu 06-28-1990
Current time is  3:39:44.97p
```

Since the files are named M0, M1, M2, etc., it is difficult to determine what message is for whom. You can use the command below to display them successively on the screen:

```
A>for %x in (m?) do type %x
```

If this command is too laborious to type, you can define a key for it. The following assigns the long command to Ctrl-F10:

```
A>prompt $e[0;103;"for %x in (m?) do type %x";13p$n$g
```

If you do not want the trouble of redefining this key repeatedly, you can put this line in F10.BAT:

```
5: echo ^[[0;103;"for %%x in (m?) do type %%x";13p
```

After defining the key either way, pressing it will display all the matching files one after another. Use Pause (or Ctrl-S) to halt the scrolling or Ctrl-C to break it.

The F10.BAT definition of the F10 key limits you to 10 message files at one time. This number should be sufficient for most circumstances. After this number is reached, MSG.BAT will no longer be able to create a file. If you find this number inadequate, you can increase it in the F10.BAT file and adjust other related items.

Whenever you are ready to print the saved messages, you can simply press Shift-F10, which is defined by F10.BAT to activate this batch file:

```
 1: @echo off
 2: rem PRINTIT.BAT
 3: echo/
 4: echo Get the printer ready and press a key to start.
 5: pause > nul
 6: echo Printing . . .
 7: copy m? prn
 8: echo/
 9: echo Press a key to delete printed files, or ^C to stop.
10: pause > nul
11: del m?
```

Line 4 displays a message and line 5 pauses the batch file; line 5 echoes the message of PAUSE to the NUL device, namely not showing it on the screen. Line 6 copies each matching file to the printer.

After the matching files are sent to the printer (or printer buffer), line 9 displays a message and lets you decide whether to delete the files, which is executed in line 11. These files named M0, M1, etc. need to be periodically deleted. If not, no file will be saved after M9 is reached.

When you have the three batch files ready, you need only manually enter this on the command line:

```
A>f10
```

This defines F10 and Shift-F10. If you want these keys automatically defined each time you power up, include the batch file lines in your AUTOEXEC.BAT file or put f10 in the last line to run (chain) F10.BAT. After these keys are defined, the rest is just a matter of pressing a single key to do a complex job.

Practice 12-6

- ☐ 1. Create the entire system described in the text, but use F9 instead of F10. Your system should enable you to press F9 to start writing a phone message, Shift-F9 to print and delete all the accumulated messages, and Ctrl-F9 to display one after another all the existing messages. (Use F12 if you have an enhanced keyboard.)
- ☐ 2. Use the completed system to simulate writing five messages. Display them and then print/delete them.
- ☐ 3. Create a separate batch file that will allow you to display one message at a time and then press a key to go to the next. Assign this batch file to Ctrl-F9.

☐ 4. If you use version 4, add to the above batch file a function that will show the name of the displayed file and allow the user to select `Y/N` to delete it. (Hint: Use `DEL /P`.)

Drill

_____ 25. The second command, entered from a batch file, will succeed in redefining the F10 key. True or false?

```
A>prompt $e[0;68;"dir"p
echo ^[[0;68;"type "p
```

_____ 26. The command `echo ^[[0;50;"MEMO";13;13;"To: "p` will assign:
a. three lines to two keys
b. three lines to one key
c. two lines to one key
d. one line to one key

_____ 27. If `ANSI.BAT` is the batch file discussed in the text (containing this key line: `echo ^[[%1;%2;%3;%4;%5)`, the command `A>ansi f` will generate an error message. True or false?

_____ 28. The command `A>echo ^V[[0;68;"dir"p` entered on the command line will work. True or false?

_____ 29. The command `A>prompt $e[1;"for %x in (*.*) do type %x";13p$n$g` will succeed in defining a key. True or false?

_____ 30. The command `echo ^[[0;50;"MESSAGE$_";13;"To: "p` contains two carriage returns. True or false?

Summary and Review

Check each item as you review and understand the following highlights of the chapter:

☐ 1. Install `ANSI.SYS` before entering ANSI escape sequences. Include the `/X` switch if you have version 4 and an enhanced keyboard. The switch allows you to redefine many more keys.

☐ 2. An ANSI escape sequence consists of Esc (ASCII 27), the left bracket ([), followed by one or more decimal numbers, and ending with a specific letter.

☐ 3. You can send an escape sequence to the computer with a number of DOS commands, the most common being `PROMPT` and `ECHO`.

☐ 4. On the command line, you can use only `PROMPT` in combination with `$e[`. The DOS prompt will disappear unless you include metastrings such as `$N$G` to restore it.

☐ 5. The most recent escape sequence or metastring sent with `PROMPT` will enter the DOS environment and stay there until another `PROMPT` command is entered. This may affect what you do in the future.

☐ 6. When `ECHO` is off, the `PROMPT` command will not work properly or immediately in sending an escape sequence. This is true regardless of whether you use `PROMPT` with `$e` or `^[` to represent the escape character.

☐ 7. Under most circumstances, it is preferable to use `ECHO` and `^[[` to enter ANSI escape sequences from a batch file. This will not affect the DOS prompt or change anything in the environment.

☐ 8. Each ANSI parameter ends with a letter. This letter must be correct in case. You cannot substitute lowercase for uppercase where the latter is required.

☐ 9. To control the cursor position or erase screen display, end an escape sequence with one of these letters: `A`, `B`, `C`, `D`, `f`, `H`, `K`, `S`, `U`, and `2J`.

☐ 10. Change screen colors with an escape sequence ending with `m` and change screen modes with `h` or `l`.

☐ 11. Redefine keys with an escape sequence ending with `p`. For single keys that appear on the keyboard, redefine them by either putting them in double quotes or entering their ASCII values.

☐ 12. To redefine extended keys, two numbers are necessary. The first is always 0 and the second is based on a number assigned to a particular key or key combination.

☐ 13. Entering an ANSI command with possibly a long string of characters can be simplified with a batch file. `ANSI.BAT`, discussed in the text, can be used to quickly enter such a command.

☐ 14. You can assign a number of keystrokes to a key to save time and speed up your work. After a key is redefined, pressing it will retrieve all the keystrokes stored there. You can use this technique to enter phrases that are repeatedly used, or quickly execute a DOS command or run a batch file.

☐ 15. The `MSG.BAT` file and related files involve extensive use of redefining some keys for quick action. They represent efficient utilization of this macro capability available in `ANSI.SYS`. Adapt them to your work with DOS and you will make yourself more productive.

Review Questions

Write an answer for each question below:

1. What are ANSI escape sequences? What does such a sequence consist of?
2. What DOS commands can be used to enter an ANSI escape sequence?
3. Describe the two methods of entering the Esc character.
4. How do PROMPT and ECHO differ in entering an ANSI escape sequence?
5. How is the DOS environment affected by ECHO or PROMPT?
6. If ECHO is turned off in a batch file, can you enter multiple PROMPT commands to enter ANSI escape sequences? Explain what will happen.
7. If ECHO is turned off on the command line, will a cursor-control PROMPT command affect the cursor behavior? Illustrate with some examples.
8. Explain what each of the following cursor-control escape sequences does:

```
A>prompt $e[H
A>prompt $e[2J
A>prompt $e[22C
```

9. What does each of the following lines do?

```
echo ^[[0;1;5m
echo ^[[0;1,5;m
```

10. What does this line do?

```
A>prompt $e[0;30;"All men are created equal";13p
```

11. What does this batch file do?

```
echo ^[[5m
echo/
echo This is a test.
echo/
echo ^[[m
```

12. What does this line do?

```
^[[97;65p^[[98;66p
```

13. What DOS commands can be used to send the above line to the computer?
14. What is the purpose of this command?

```
A>prompt $e[0;68;"memo m0 m1 m2 m3 m4 m5 m6 m7 m8 m9";13p
```

15. What does this line do?

```
echo ^[[0;32;"echo|more|date >> notes";13p
```

Version Comparison Chart

Commands/Features	Version 3.3	Version 4
`ANSI.SYS`	Can't define F11, F12, and some other keys on an enhanced keyboard	Can define F11, F12, and other keys if `/X` is used in installing the driver

Appendix A

Glossary

(Note: If a term appears in **boldface**, it is explained separately.)

▼ A ▼

Adapter See controller card.

Allocation unit See cluster.

Alphanumeric characters **ASCII** characters, which may be alphabetic or numeric. Each one, letter or number, has a specific value; see Appendix B.

ANSI American National Standard Institute. An organization that determines standards for various computer-related activities, including the **ANSI escape sequence** and **ASCII**.

ANSI escape sequence An ANSI command to extend PC input and output, including screen colors and keyboard redefinition. To use an ANSI escape sequence, you must first install `ANSI.SYS`.

`ANSI.SYS` A special file that comes with a DOS package. This file must be included in your `CONFIG.SYS` file before DOS can implement an ANSI escape sequence.

Application software A computer program intended to handle a specific task. Examples include word processors, spreadsheet programs, and database management systems. It is often contrasted with **system software**.

Archive An attribute given to a file. A file normally carries an archive bit. This attribute is shown when you use the `ATTRIB` command to display a file. Some commands such as `BACKUP` and `XCOPY` can maneuver this archive bit in backing up a file. `ATTRIB` can add or remove this archive bit.

Argument Also known as command-line argument; see parameter.

ASCII American Standard Code for Information Interchange, a common standard for converting characters to numerical values, which a computer can understand and maneuver. There are 256 ASCII characters. The first half (0–125), including all the **alphanumeric characters**, are standardized. The second half (126–255), known as extended ASCII characters, contain various drawing characters and are not standardized. The table in Appendix B shows the extended characters adopted by IBM.

ASCII file A file that contains only ASCII characters. A file created with `COPY CON` and EDLIN is an ASCII file. Various DOS commands can handle such a file. A file created with an application program such as a word processor usually contains various formatting characters and is thus not a pure ASCII file. Most word processors, however, allow you to create an ASCII file.

AT Originated from IBM PC-AT. AT stands for Advanced Technology. The original IBM AT, introduced in 1984, was based on the Intel 80826 CPU. Today, an AT-class PC can be any non-IBM computer based on 80286, 80836, or 80846 CPU. IBM's new products are renamed **PS/2 and PS/1.**

`AUTOEXEC.BAT` Automatic executable batch file. It can contain various DOS commands, such as setting DOS paths and loading a program. It must have the exact name and be stored on the booting disk. After DOS is booted, it looks for `CONFIG.SYS`, if any, to execute, and then this **batch file**, if any.

`AUX` AUXiliary device or **serial port**; can be `COM1`, `COM2`, `COM3`, or `COM4` (versions before 3.3 can handle only `COM1` and `COM2`).

▼ B ▼

Batch file A user-created file that can contain a series of DOS commands. When a batch file is executed, all the commands in it will be executed one by one. A batch file can be written to behave like a simple computer program, with branching, looping, and other techniques. A batch file must be **ASCII** text and have a `BAT` extension in its file name.

Baud Transmission rate for **modem** and other devices. A baud is 1 **bit** per second (bps). Most modems sold today are 2400-baud models.

Binary A base 2 numerical system. A binary system has only two digits, 0 or 1.

BIOS Basic Input-Output System. Instructions (computer code) stored in a ROM chip to control input and output devices. BIOS is the first step in booting DOS.

Bit A binary digit. A bit is either 0 (off) or 1 (on). This is the smallest unit a computer can maneuver.

Boot record A disk record (program) created by formatting. This record is used to load a system (hidden) file in the booting process.

Boot disk Or bootable disk. A disk containing two hidden system files and `COMMAND.COM`, which can be used to start DOS. The `/S` switch of `FORMAT` can be used to create a boot disk. The `DISKCOPY` and `SYS` commands can transfer the hidden files to a new disk to make it bootable. Other commands do not recognize the hidden files.

Boot To start DOS or to make it pull itself up from its bootstraps.

Branching A programming technique that forces program or batch execution to interrupt the top to bottom routine and go to another routine to continue execution. In batch files, `GOTO` is used for unconditional branching and `IF-GOTO` for conditional branching.

Buffer A memory area for storing temporary data. The `BUFFERS` command in a `CONFIG.SYS` file can increase the default of 2 or 3 disk buffers to an optimal number of 20–30. A DOS or EDLIN command is also stored in a buffer called a **template**.

Bug An error in a computer program or **batch file.**

Bus Also known as data bus. Circuitry inside a PC used to transmit information from one part to another. See MCA and EISA.

Byte 8 **bits** constitute 1 byte. A byte is used to form a character. If a file contains a single character and nothing else, it is 1 byte long.

▼ C ▼

CD-ROM Compact Disk–Read-Only Memory. A **ROM** compact disk resembles an audio CD and can store huge amounts of data. The data stored on a ROM disk cannot be erased. A special player is needed to retrieve stored data.

CGA Color Graphics Adapter, IBM's first color graphics card used to run an **RGB** monitor.

Chaining A programming technique that transfers control from one program to another without returning to the first program. See nesting.

Chip A small device made of silicon containing integrated circuits. Chip examples include **RAM, ROM,** and **CPU.**

Clones See compatibles.

Cluster Also known as allocation unit. A cluster is the smallest disk space DOS can work with. The size of a cluster varies from one type of disk to another. On a floppy disk, a cluster can be 1 or 2 **sectors,** namely 512 **bytes** or 1**KB**. A file as small as 1 byte can thus take up as much space as 1KB because DOS is not capable of utilizing a smaller disk space.

Code page A set of special characters belonging to a foreign language. If you want DOS to handle some unique characters belonging to a foreign language, you must tell it to use another code page number.

`COM1` The first serial communications port. Version 3.3 and later can handle up to `COM4`; those before, only `COM1` and `COM2`.

Command See DOS command.

Command line The line on the monitor screen where a DOS command (instruction) is to be entered.

Compatibles PCs that are compatible with IBM models. An IBM-compatible PC can use parts and run software intended for an authentic IBM PC.

Computer virus A computer program that is designed to damage computer files.

CON Short for console, which is a combination of keyboard and monitor. CON is recognized by DOS as a device representing standard input (keyboard) and standard output (monitor).

CONFIG.SYS A file you create to define various PC hardware. This file must be put on the booting disk. As soon as DOS is booted, it looks for this file, if any, to execute. Common directives and driver files included in a CONFIG.SYS file are BUFFERS, FILES, ANSI.SYS, and VDISK.SYS (or RAMDRIVE.SYS).

Control characters The first 32 (0–31) ASCII characters. These are used to control computer hardware and software.

Controller card A computer device inserted into an expansion slot and connected to other devices such as a printer and disk drive.

CP/M Control Processor/Micro, a popular operating system for micros before DOS. It has largely been displaced by DOS.

CPU Central Processing Unit. The brain of a computer, the CPU performs arithmetic and logic functions, and controls other parts of a computer. Most PCs today use a line of Intel CPU chips ranging from 8086/88 to 80486.

Current directory The active directory, the place DOS is supposed to access when not directed to do otherwise.

Cursor A little blinking underscore character on the screen to designate the location where a character is to be displayed.

Cylinder A track of a hard disk platter.

▼ D ▼

Daisy-wheel printer A printer using a daisy wheel to produce printed characters. A daisy wheel resembles the typing elements (keys) of an electric typewriter. Such a printer produces fully formed characters, but is not capable of printing graphics. Once popular, daisy-wheel printers have mostly been replaced by dot-matrix printers.

DAT Digital Audio Tape, a magnetic tape resembling audio cassette tape. It can store huge amounts of data and also provide random access like a diskette.

DEBUG A DOS program that can be used to write small assembly programs, or identify and correct errors in a program.

Debug Identify and correct errors, an essential and tedious process in writing computer programs.

Default drive The active disk drive, whose letter is shown in the DOS prompt, such as A> or C>.

Device A piece of computer equipment that DOS can recognize and maneuver. DOS recognizes it via reserved words such as AUX, COM1, CON, and PRN.

Device driver A compiled computer file intended to maneuver a computer device. A device drive, once installed, can extend DOS's ability to handle an unusual

computer device. The DOS package contains a number of such drivers, which usually have the file name extension of SYS.

Directory An indexing system used by DOS to manage files. A directory can have a parent directory, one level above it; or one or more subdirectories, one level below it. Each disk has a root directory, which is created by DOS in the formatting process. Other directories may be created or removed by the user.

Directory entries The information related to files and directories. The DIR command is used to show directory entries.

Disk Magnetically coated storage media. Two commonly available types are floppy and hard (fixed) disks.

Disk buffer See buffer.

Disk drive A device used to maneuver disks for storing and retrieving data.

DOS command An instruction you can give which DOS can understand and execute. There are several dozen DOS commands; some are **internal** and others are **external.**

DOS device name A special term treated by DOS as a computer device. **PRN**, for example, is considered to be a printer. See reserved words.

DOS editing keys Certain keys on the PC keyboard used for editing DOS and EDLIN commands. These include Esc, Ins, Del, Backspace, and some function keys.

DOS environment A memory area set aside for DOS to maneuver certain data. Normally, PATH, PROMPT, and APPEND (if installed with /E) strings are entered into the environment. These strings determine how DOS is to display its system prompt or search for executable and data files. The environment has limited space; it can be expanded if necessary. See environment variable.

DOS prompt Also known as system prompt. It marks the location where a DOS command is to be entered. The DOS prompt by default contains the letter of the default drive and the > sign. It can be changed with the PROMPT command.

Dot-matrix printer A printer that prints characters by using a matrix of pins (wires). Currently 9-pin and 24-pin models are common. The more pins a printer has, the more fully formed characters it can produce.

Dummy variable See replaceable parameter.

▼ E ▼

EGA Enhanced Graphics Adapter. An EGA card is needed to drive an EGA monitor. The duo was introduced with the AT in 1984. While still popular, the EGA standard is gradually being replaced with **VGA.**

EISA Extended Industry Standard Architecture. This is a bus structure designed by a consortium of IBM rivals to counter IBM's **proprietary MCA** bus. A PC based on this bus can utilize existing AT parts while an MCA-based PC cannot.

Enhanced keyboard A keyboard introduced with the **PS/2** line of PCs. It has 101 keys and 12 function keys. An additional cursor pad is also added.

Environment See DOS environment and operating environment.

Environment variable Also known as named parameter. An alphanumeric string entered into the **DOS environment**. Use `SET` to assign a value to a variable and enter both into the environment. Use a batch file to maneuver environment variables enclosed in a pair of `%` signs, such as `%var%`.

Errorlevel Used with `IF` in a **batch file** to respond to an **exit code**. For example, if `BACKUP` cannot find a source file to back up, it reports errorlevel 1. You can thus use a batch file to respond to this situation. DOS also treats a character as an errorlevel number identical to the character's ASCII value. This phenomenon can be used to construct a menu batch file.

Executable file A file DOS can execute. It could carry a `COM`, `EXE`, or `BAT` extension. If there are three files with one file name but with these separate extensions, the `COM` file has priority over the `EXE` file, which has priority over the `BAT` file.

Exit codes Numbers reported by DOS in response to the outcome of some commands. `FORMAT`, for example, can output exit codes depending on whether formatting is successful. These codes can be maneuvered with an **errorlevel** in a batch file.

Expanded memory Additional memory installed on any PC. Memory chips are installed on an expansion board, which is inserted into an expansion slot. Boards conforming to the LIM (Lotus-Intel-Microsoft) specifications can be used by many DOS-based application programs. DOS can use up to only 640KB of conventional memory, but it can take advantage of as much as 8MB of expanded memory.

Expansion slot A slot on the **motherboard** to which a **controller card** can be inserted. The controller card can then be connected to a device such as a printer or disk drive.

Extended key codes Used in `ANSI.SYS` to redefine keyboard keys. Some keys, such as function keys, require two numbers (the first number is always 0) if you want to redefine them. These numbers are known as extended key codes.

Extended memory Additional memory installed on a PC based on 80286 CPU or higher. DOS cannot use more than 640KB of such memory. OS/2, UNIX, and Windows can use such memory up to 15MB. Some programs (such as Quarterdeck's QEMM) can convert the extra memory to **expanded memory**.

External command A DOS command that is not included in `COMMAND.COM`. It is stored on a disk. To use such a command, it must be located in the **current directory** of the **default drive**, or DOS must be told where to access it.

▼ F ▼

FAT File allocation table, a table created by formatting and used by DOS to keep track of disk space usage, and find files to read and space to store data. Two FATs are maintained on each disk.

File A collection of related data. Each file has a unique name. DOS recognizes some files by their extensions, such as `COM`, `EXE`, `BAT`, and `SYS`.

File handles The number of files DOS can handle at one time. The FILES command included in a `CONFIG.SYS` file can be used to increase the default of 8 file handles to an optimal number of 20–30.

Filter A DOS command that can sieve through information going to the screen and produce output in a specified way. `FIND`, `MORE`, and `SORT` are DOS filter commands.

Firmware A device that is a combination of software and hardware. It is usually a **chip** (which is hardware) containing computer instructions (software). When power is on, the instructions are acted upon; when power is off, they do nothing. A **BIOS** chip is an example of firmware.

Function keys The keys on the PC keyboard marked as F1, F2, etc. DOS uses only the first 6 keys for editing. If `ANSI.SYS` is installed, these keys and their various combinations can be defined for your own use.

▼ G ▼

Gigabyte One billion or 2^30 bytes.

▼ H ▼

Hercules A monographics card allowing graphics to be displayed on a monochrome monitor. Most mono monitors today are equipped with cards that are compatible to the Hercules standard.

Hexadecimal A base 16, or 2-byte, numerical system. Hex digits consist of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Hex is often used in writing computer programs. `DEBUG`, for example, can recognize only hex numbers.

▼ I ▼

Index hole A round hole near the center of a 5.25-inch disk to control the disk's reading and writing. On a 3.5-inch disk, a rectangular hole on the back of the disk called a sector notch serves the same purpose.

Installable Device Driver See device driver.

Internal command A DOS command included in `COMMAND.COM`. When DOS is booted, `COMMAND.COM` stays in the memory. Since an internal command is in **RAM**, it can be used regardless of what the default drive or current directory is. DOS does not need to go to a disk to access it.

▼ K ▼

KB Kilobyte, 1024 or 2^10 bytes.

▼ L ▼

Label A marker for branching batch execution. A label in a batch file is a text string preceded with a colon (:).

Laser printer A printer using a laser beam and photocopy technology. It produces printout by using a large number of dots. Today, the low-end models output 300 by 300 dots per inch. High-end models used in typesetting have much higher resolutions.

Logical drive A drive that DOS can recognize and manipulate. Normally DOS can handle four drives, A, B, C, and D. If you want it to handle more, you need to put the `LASTDRIVE=x` command in your `CONFIG.SYS` file. After that, X becomes the last (highest) drive DOS can maneuver. If a logical drive is not paired with a physical drive, an error may result.

Looping A programming technique that forces execution of a routine a number of times. The `FOR..IN..DO` command is the most commonly used looping command.

LPT1 The (line) printer connected to the first **parallel port.**

▼ M ▼

Macro An arrangement that allows a series of keystrokes to be stored in one key. When that key is pressed, all the stored keystrokes will be displayed on the screen. You need to install `ANSI.SYS` to do this with DOS.

Mainframe A large-scale computer.

MCA Micro Channel Architecture. This is a **proprietary** bus structure introduced with the **PS/2** line of PCs. An AT add-in card cannot be plugged into the motherboard of this system. See EISA.

MDA Monochrome Display Adapter, the card used to drive IBM's mono monitor. This system produces crisp screen characters, but cannot handle graphics. It is mostly displaced by **Hercules** cards, also known as MGA (monographics adapter).

Megabyte One million or 2^20 bytes.

Memory See RAM.

Menu A screen display of options available in a computer program or batch file.

Metastring A text string placed after the `PROMPT` command to maneuver the DOS prompt. Such a text string usually consists of a special character put after a $ sign.

Microcomputer Or micro. A tiny computer. This term was popularly used in the 1970s to mean any computers smaller than the minicomputer. Today it is mostly used for a small computer primarily for playing games.

Minicomputer Or mini. A small-scale computer. It is between a **mainframe** and a PC. Most minis use **UNIX** as the **operating system.** Today's high-end (386 and 486) PCs are replacing minis.

Modem MOdulator DEModulator. A communications device used to send data between a PC and another PC, a bulletin board, or a commercial database like CompuServe. Modem speeds are measured in **bauds.** When modems first became popular, 300-baud models were common. They were quickly replaced by 1200-baud models. Today, 2400-baud models are most common.

Motherboard Also known as main board or system board. A flat device placed at the bottom of a **system unit**. All the vital parts of a PC, such as **CPU, RAM, controller cards**, etc. are placed on it or connected to it.

Mouse An input device that resembles an animal of the same name. A mouse can be used to control the cursor, maneuver objects on the screen, or execute a program. Character-oriented programs generally have little use for this device, but graphics-oriented programs often cannot be run without it.

MS-DOS Microsoft Disk Operating System. A DOS marketed by a vendor other than IBM is generally known as MS-DOS. MS-DOS and PC-DOS are practically identical.

Multitasking Doing several jobs at the same time. **UNIX** and **OS/2** are examples of multitasking **operating systems**.

▼ N ▼

Nesting A programming technique that branches execution to a routine and then returns. In a **batch file**, nesting means using the `CALL` command to execute another batch file and then returning to the original batch file. See chaining.

NLQ Near Letter Quality, said of print quality. Some **dot-matrix printers** allow you to select this mode to produce more fully formed characters.

▼ O ▼

Octal A base 8 numerical system. Its digits range from 0 to 7.

Operating environment A kind of computer program that provides an interface between the user and compatible application programs. Some can be character-based and others graphics-based. The Shell program in the WordPerfect Library is an example of the former. Microsoft Windows, a graphical operating environment resembling the Mac, has become a popular package in recent years.

Operating system Or OS. A computer program that maneuvers mostly computer hardware such as CPU, RAM, printer, and monitor. DOS, **OS/2**, and **UNIX** are popular operating systems for the PC.

OS/2 Operating System/2. A new operating system created by Microsoft. It can be run only on a PC with 80286 or higher **CPU**. It is primarily a multitasking operating system. With additional equipment and software, it can also be a multiuser system like UNIX.

▼ P ▼

Parallel port An interface for a parallel device, such as a printer. A parallel device, unlike a serial device, transmits data 8 bits (1 byte or character) at a time.

Parameter An argument or qualifier that can be used with a DOS command or is expected by a batch file. To copy a file, for example, you need to have a **source** and a **target**; these are considered parameters. A batch file may also expect one or more files; these are also parameters.

Parent directory A **directory** one level above the current one.

Partition A unit in a hard disk system. The `FDISK` command is used to create partitions. Before a hard disk can be used by DOS, it must be partitioned and then formatted.

Path The specific trail that DOS is supposed to search for a file to execute or to read. A predetermined path can be entered into the **DOS environment** by using the `APPEND` or `PATH` commands.

Path name A combination of directory path and file name, such as:

```
A>c:\dos\format a:
```

PC Personal Computer. It originated from the IBM PC. Today it has become a generic term applied to any serious computer smaller than the **minicomputer**. Subcategories include **XT, AT, PS/2, PS/1**, etc. Most PCs use Intel CPUs and are compatible with IBM models.

PC-DOS Personal Computer Disk Operating System. A DOS marketed by IBM is known as PC-DOS. Those marketed by other vendors are known as MS-DOS. Both are practically identical.

Peripheral An external computer device, such as a printer.

Pipe A connector of two DOS commands. The ¦ sign is used to pipe two commands, such as:

```
A>dir ¦ more
```

Pixels Dots displayed on the screen. Characters and graphics images are formed with pixels. The more pixels, the better-looking the display.

Platter A disk inside a hard drive. A hard disk unit usually contains a number of platters.

Print queue A list of files lined up to be printed one after another. The `PRINT` command can add files to or delete them from a print queue.

Printout A printed (hard) copy of a **file**.

PRN A **reserved word** treated by DOS as the standard printer.

Prompt A cue or signal for user response. The **DOS prompt** invites you to enter a command. Some DOS commands prompt you to swap disks and others to enter Y or N.

Proprietary Controlled by one company, as a contrast to open standard. Large vendors like IBM, Apple, and Compaq like to sell proprietary products because they are highly profitable and competitors cannot market compatible products. Consumers, however, like open-standard products because they have more choices of good-quality, low-priced parts that are interchangeable.

PS/1 Personal System/1, a line of PCs introduced by IBM in 1990. Intended for the home market like the PCjr, these models use the 80286 CPU and the traditional AT bus.

PS/2 Personal System/2, a line of small IBM computers introduced in 1987. It replaces the original PC-XT and PC-AT. This new line is a hodge-podge combination of numerous products based on a variety of CPUs ranging from 8086 to 80486. The low-end products, based on 8086 CPUs, are similar to the original PC or XT.

Most of the higher-priced models use 80286 or higher CPUs and are based on a new **proprietary bus** structure called **MCA**. Rival vendors have produced models based on the extended AT bus structure called **EISA**.

▼ Q ▼

Queue See print queue.

▼ R ▼

RAM Random Access Memory, or computer memory. It's called random access because any part of the memory can be accessed at will. RAM is volatile. When power is on, it can be used; when power is off, the data in it are erased.

RAM disk A memory area set aside to simulate a real disk. A RAM disk (or drive), also known as a virtual disk, can temporarily store data. When power is off, the stored data are wiped out.

Read-only An attribute marking a file. Such a file can only be read but cannot be altered or erased. The `ATTRIB` command is used to add or remove this attribute.

Read-write opening A disk area that is opened for reading and writing. On a 5.25-inch disk, this area is exposed. On a 3.5-inch disk, it is covered with a shutter, which is automatically opened when a disk is inserted into a drive.

Redirection Redirecting input or output from the standard device. Normally, DOS expects an input from the keyboard when you issue a command like this:

```
A>del *.*
```

You can, however, supply the expected input of Y or N from a file. A normal output also goes to the screen. It can also be redirected. This command redirects output to a disk file:

```
A>dir > dirfile
```

Replaceable parameter A batch file parameter (argument) that can behave like a variable. It is represented with `%`, followed by a number, such as `%1`. A batch file name is considered parameter 0, or `%0`; the first item after the file name is treated as parameter 1, or `%1`; and so on. These parameters can be maneuvered with the `SHIFT` command. You can use the `if x==x%1` equation to test whether an expected parameter is entered with a batch file.

Reserved words Words that are used by DOS (or another computer language) for special purposes. Such words, including `AUX` (serial port), `COM1` (1st serial port), `CON` (console), `NUL` (dummy, nonexisting, device), `PRN` (parallel port), and `LPT1` (1st parallel port), cannot be used as file names.

RGB Red, Green, and Blue, used to designate the first color monitor marketed by IBM. To run an RGB monitor, you need to equip your PC with a **CGA** graphics card. The combination produces dotty characters and crude graphics. Today, many low-end PCs still use this combination; more expensive models, however, are equipped with **EGA** or **VGA** systems.

ROM Read Only Memory. It is usually a chip containing computer instructions that, unlike **RAM**, cannot be erased. A PC includes a number of ROM chips, one of which contains **BIOS**.

Root directory The highest-level **directory**. Every disk, even though not divided into any subdirectory, has a root directory, which is created by `FORMAT`.

▼ S ▼

Scanner A device that can read printed text or graphics and feed the data into an application program. Some models are equipped with software that can recognize and maneuver printed words; others simply treat any scanned object as an image that cannot be maneuvered.

Sector A subunit of a **track**. Each sector has the storage capacity of 512 bytes. A track can be divided into 8 or 9 sectors for a 5.25-inch, double-density disk; 15 sectors for a 5.25-inch quad-density disk; 9 sectors for a 3.5-inch, double-density disk; and 18 sectors for a 3.5-inch, quad-density disk.

Sector notch See index hole.

Serial port An interface for a serial device, such as a mouse or modem. A serial device transmits data one bit at a time, in contrast to 8 bits for a **parallel device**.

Source The original object for DOS to act on. Some commands, `COPY` for example, expect a source and a target.

Standard input The device where raw data are normally entered. The keyboard is the standard input device.

Standard output The device to which computer output is normally sent. The screen is the standard output device.

Subdirectory A **directory** that is one level below the current one.

Surge protector A device to control a sudden surge in the electrical current and prevent it from harming your computer. It usually comes with an on-off switch and multiple electrical outlets.

Switch An option used with certain DOS commands. To format a bootable disk, for example, you need to use the `/S` switch.

System prompt See DOS prompt.

System software See operating system.

System unit A rectangular metal box containing the vital parts of a PC, including the **motherboard, CPU, ROM, RAM, disk drives**, etc.

▼ T ▼

Target Opposite of **source**. A target drive of the `COPY` command, for example, is the place to which duplicates from a source drive are sent.

Template A memory area (**buffer**) where the most recently entered DOS (or EDLIN) command is stored. This stored line can be copied to the screen and maneuvered with **DOS editing keys**.

Toggle Used with certain keys for switching between two modes. The Ins key, for example, is a toggle key. It can toggle (switch) between insert and typeover modes. Normally this key operates in insert mode. Pressed once, it goes to typeover mode. Pressed another time, it goes back to insert mode, and so on.

Track A concentric ring of a disk created by formatting. Currently, a floppy disk can be divided into 40 (5.25-inch double-density) or 80 (5.25-inch quad-density and 3.5-inch) tracks.

TSR Terminate and Stay Resident. Some computer programs are memory resident once loaded. You can press a certain key combination to activate it. When these keys are not pressed, the program stays dormant.

▼ U ▼

UNIX A multiuser **operating system** initially developed by Bell Labs of AT&T. It is a popular operating system with minicomputers and widely used in colleges. It is being adapted to run on 386/486 PCs.

▼ V ▼

VGA Video Graphics Array. A video card introduced with the **PS/2**. It is used to drive a VGA monitor. This duo is the current video display standard, providing displays that are more attractive than previous systems.

Video graphics card A controller card inserted into an expansion slot to control a monitor. The most common cards today are **Hercules, CGA, EGA,** and **VGA.**

Volume label An optional text string (up to 11 characters) you can use to identify a disk. `FORMAT`, `LABEL`, and `VOL` can create and display volume labels.

▼ W ▼

Wild card characters The ? and * characters. A question mark is used to represent any character in a particular position in a file name. An asterisk can represent any number of characters from a certain position.

Write-protect notch/switch A device on a floppy disk to prevent writing to a write-protected disk. On a 5.25-inch disk, this is a cut-out rectangular hole. When this hole is covered, the disk is write-protected. On a 3.5-inch disk, a built-in switch can be pushed to the edge, thus showing a see-through hole, to write-protect the disk.

▼ X ▼

XT Extended Technology. It originated from the IBM PC-XT. Today an XT-class PC is an entry-level PC equipped with 8086/8088 **CPU.**

Appendix B

ASCII Characters

0		43	+	86	V	129	ü	172	¼	215	╫
1	☺	44	,	87	W	130	é	173	¡	216	╪
2	☻	45	-	88	X	131	â	174	«	217	┘
3	♥	46	.	89	Y	132	ä	175	»	218	┌
4	♦	47	/	90	Z	133	à	176	░	219	█
5	♣	48	0	91	[	134	å	177	▒	220	▄
6	♠	49	1	92	\	135	ç	178	▓	221	▌
7	•	50	2	93	]	136	ê	179	│	222	▐
8	◘	51	3	94	^	137	ë	180	┤	223	▀
9	○	52	4	95	_	138	è	181	╡	224	α
10	◙	53	5	96	`	139	ï	182	╢	225	ß
11	♂	54	6	97	a	140	î	183	╖	226	Γ
12	♀	55	7	98	b	141	ì	184	╕	227	π
13	♪	56	8	99	c	142	Ä	185	╣	228	Σ
14	♫	57	9	100	d	143	Å	186	║	229	σ
15	☼	58	:	101	e	144	É	187	╗	230	µ
16	►	59	;	102	f	145	æ	188	╝	231	τ
17	◄	60	<	103	g	146	Æ	189	╜	232	Φ
18	↕	61	=	104	h	147	ô	190	╛	233	θ
19	‼	62	>	105	i	148	ö	191	┐	234	Ω
20	¶	63	?	106	j	149	ò	192	└	235	δ
21	§	64	@	107	k	150	û	193	┴	236	∞
22	▬	65	A	108	l	151	ù	194	┬	237	φ
23	↨	66	B	109	m	152	ÿ	195	├	238	ε
24	↑	67	C	110	n	153	Ö	196	─	239	∩
25	↓	68	D	111	o	154	Ü	197	┼	240	≡
26	→	69	E	112	p	155	¢	198	╞	241	±
27	←	70	F	113	q	156	£	199	╟	242	≥
28	∟	71	G	114	r	157	¥	200	╚	243	≤
29	↔	72	H	115	s	158	₨	201	╔	244	⌠
30	▲	73	I	116	t	159	ƒ	202	╩	245	⌡
31	▼	74	J	117	u	160	á	203	╦	246	÷
32		75	K	118	v	161	í	204	╠	247	≈
33	!	76	L	119	w	162	ó	205	═	248	°
34	"	77	M	120	x	163	ú	206	╬	249	∙
35	#	78	N	121	y	164	ñ	207	╧	250	·
36	$	79	O	122	z	165	Ñ	208	╨	251	√
37	%	80	P	123	{	166	ª	209	╤	252	ⁿ
38	&	81	Q	124	|	167	º	210	╥	253	²
39	'	82	R	125	}	168	¿	211	╙	254	■
40	(	83	S	126	~	169	⌐	212	╘	255	
41	)	84	T	127	⌂	170	¬	213	╒		
42	*	85	U	128	Ç	171	½	214	╓		

(Note: 0 is a null character; 32 is a space, the same as pressing the space bar; 255 is a blank. To enter a character, hold down Alt and type a corresponding number on the numeric keypad. 1–31 are not supported by DOS; e.g., Alt-1 or Ctrl-A shows ^A, not ☺.)

Appendix C

DOS Versions

DOS versions are usually shown with two- or three-digit numbers. The first digit indicates a major release, and the second a minor revision. The third digit, if any, signifies a bug-fix release (e.g., 4.01) or a special version (e.g., 2.11) for a less compatible system such as PC Junior and Sanyo MBC-550. Use the VER command to show what DOS version you are currently using.

Each new version provides a few new commands to enable DOS to perform more sophisticated tasks. Some old commands also have new added functions to handle newly available hardware such as a hard disk and a 3.5-inch floppy disk.

The list below shows each version's new commands (and device driver files) and the old commands' enhancements. If a command provides no explanation, related information is available in Appendix D.

Version 1

Original Commands

```
CHKDSK
COMP
COPY
DATE
DEBUG
DEL (ERASE)
DIR
DISKCOMP
```

EDLIN
FORMAT (initially one side, and then two sides up to 320KB)
MODE
PAUSE
REM
REN
SYS
TIME
TYPE

Version 2

New Commands and Drivers

ANSI.SYS
ASSIGN
BACKUP
BREAK
CD (CHDIR)
CLS
COMMAND (to load another copy of COMMAND.COM)
CTTY (to change standard input/output device)
DEVICE (configuration command to specify a driver file)
DISKCOPY
ECHO
EXIT (to exit an extra copy of COMMAND.COM)
FC (MS-DOS version of COMP)
FDISK
FILES
FIND
FOR
GOTO
IF
GRAPHICS
MD (MKDIR)
MORE
PATH
PRINT
PROMPT
RD (RMDIR)
RECOVER
RESTORE
SET
SHELL (to load another command processor)
SHIFT

SORT (no adjustment for uppercase and lowercase, placing "Z" before "a")
TREE
VER
VERIFY
VOL

Enhancements

This version can format a 5.25-inch floppy disk up to 360KB.
Support for hard disks and installable device drivers.
Support for piping (|), filtering (FIND, MORE, SORT), and input-output redirection (<, >, >>).
EDLIN: C, M, P, and T commands added.

Version 3.O

New Commands and Drivers

ATTRIB (to add or remove only R attribute)
COUNTRY (configuration command for setting country-specific date-time format)
FCBS (configuration command for specifying the number of file control blocks that can be currently open)
KEYBxx (to set country-specific keyboard layout)
GRAFTABL (to display some characters on CGA monitor)
LABEL
LASTDRIVE
SELECT (for country-specific keyboard layout and date-time format)
SHARE (for file sharing and locking)
VDISK.SYS (configuration command to create RAM disks)

Enhancements

A directory path can be specified before an external command, such as:

```
A>c:\dos\format a:
```

FORMAT: can format a 1.2MB floppy disk; has a new /4 switch to format a 360KB disk in a 1.2MB drive; a new warning message for formatting a hard disk.
BACKUP and RESTORE: can back up files to a hard disk or from a floppy to another floppy.
DISKCOMP and DISKCOPY: can handle 1.2MB disks.
DATE: can support mm-dd-yy or dd-mm-yy formats.
GRAPHICS: can support color printing.
SORT: uppercase and lowercase adjusted, placing "a" before "Z".

Version 3.1

New Commands and Drivers

JOIN
SUBST

Enhancements

This version begins support for networking.
LABEL: a new prompt before deleting a volume label.
SHELL: a new switch of /E:size to expand the size of the environment;
size is the number of 16-byte blocks.
TREE: a new /F switch to display files.

Version 3.2

New Commands and Drivers

DRIVER.SYS (to permit assigning a letter to a floppy drive)
RAMDRIVE.SYS (MS-DOS version of VDISK.SYS)
REPLACE
XCOPY

Enhancements

ATTRIB: a new option to set or remove the A attribute.
FORMAT: can format 3.5-inch, 720KB disks.
DISKCOMP and DISKCOPY: can handle 720KB disks.
SHELL: the size in the /E:size switch is changed to bytes.

Version 3.3

New Commands and Drivers

APPEND
CALL
CHCP (to change code pages)
DISPLAY.SYS (to display language-specific characters)
FASTOPEN (to speed up disk access)
NLSFUNC (national language support function; to specify a country driver
file before changing code pages)
PRINTER.SYS (to print language-specific characters)

Enhancements

Support for 1.44MB disks and up to four (instead of two) serial ports (COM4).
ATTRIB: /S switch added.
BACKUP: new switches of /F, /L, and /T; target disk need not be preformatted.
Batch files: @ to suppress batch execution and %var% to maneuver environment variables; CALL available for the first time to replace COMMAND.COM/C.
DATE/TIME: changes the system's calendar.
FDISK: divides a large hard disk (more than 32MB) into multiple drives.
FORMAT: new switches of /N and /T.
MODE: can handle code pages.
KEYB: replaces KEYBxx of version 3.2.
RESTORE: new switches of /B, /E, /L, and /N.
SHELL: default environment size increased from 128 to 160 bytes.

Version 4

New Commands and Drivers

MEM
SHELL.BAT (to start DOS Shell environment)
SMARTDRV.SYS (disk cache)
XMAEM.SYS (to use 386 extended memory to simulate expanded memory)
XMA2EMS.SYS (to use LIM expanded memory)

Enhancements

Disk serial numbers are created and displayed by various commands.
A new (optional) user interface that resembles Microsoft Windows.
ANSI.SYS: new switches of /X, /K, and /L for enhanced keyboard and EGA/VGA monitors.
APPEND: adds new switches of /X:ON, /X:OFF, /PATH:ON, /PATH:OFF.
BACKUP: needs access to FORMAT.COM and automatically formats target disk if necessary.
CHKDSK and FORMAT: show allocation unit (cluster) information.
DEL (ERASE): a new /P switch.
FDISK: can create a larger partition.
FORMAT: adds /F and /V:diskname switches.
MODE: can speed up keyboard and increase display lines for EGA and VGA.
REPLACE: a new /U switch.
REM: remarks permitted in a CONFIG.SYS file.
SYS: can specify a source drive other than where SYS.COM is stored.
TIME: can use 24-hour format or "a" or "p" in 12-hour format.
TREE: displays diagram (not list); has a new /A switch; can analyze a directory as well as a whole disk.
XCOPY: no longer returns exit codes.

Appendix D

DOS Commands

Quick Reference

B = batch (internal) command
C = configuration (internal) command
D = device driver file
E = external command
I = internal command
U = user-created

Command		Purpose
ANSI.SYS	D	Special driver to enhance input/output
APPEND	E	To set/display path for text files
ASSIGN	E	To assign one drive to another
ATTRIB	E	To set or remove A and R file attributes
BACKUP	E	To back up files; used with RESTORE
BATCH	U	To store and execute a series of commands
BREAK	CI	On/off to determine DOS's check on ^C
BUFFERS	C	To set the number of memory buffers
CALL	B	To call (nest) another batch file
CD	I	To change to another directory
CHKDSK	E	To check a disk for various information
CLS	I	To clear the screen
COMP	E	To compare files (PC-DOS only)
CONFIG.SYS	U	To configure (define) the system
COPY	I	To copy, combine, and create files

Command		Purpose
COPY CON	I	To create files (copy from console to disk)
DATE	I	To display and/or change PC clock date
DEL	I	To delete file(s)
DIR	I	To show disk directory information
DISKCOPY	E	To duplicate a disk (copy by tracks)
ECHO	BI	To display ECHO on/off or batch message
EDLIN	E	A line editor to create text files
ERASE	I	To erase files, same as DEL
EXIST	B	To test existence of a file or error level
FC	E	To compare files (MS-DOS only)
FDISK	E	To partition a hard disk
FILES	C	To set the number of file handles
FIND	E	To find text lines matching a search string
FOR	I	To loop a command a number of times
FORMAT	E	To prepare a disk for use
GOTO	B	To branch to a batch label
GRAPHICS	E	TSR, to print a screen graphics display
IF	B	To test parameter, file, and error level
JOIN	E	To join a directory to an empty drive
KEYS	-	Keys commonly used by DOS
LABEL	E	To display/change disk volume label
LASTDRIVE	C	To set the last logical drive
MD	I	To make (create) a directory
MEM	E	To show memory use
MODE	E	To check/set monitor, printer, and keyboard
MORE	E	To filter text lines
PATH	I	To display/set path for executable files
PAUSE	B	To pause batch execution
PRINT	E	To print text files in the background
PROMPT	I	To set prompt, enter metastring and ANSI commands
RD	I	To remove directory
RECOVER	E	To recover files from an unreadable disk
REM	B	To add remarks in a batch file
REN	I	To rename a file
REPLACE	E	To replace (update) files
RESTORE	E	To restore files created by BACKUP
SET	I	To set/display environment variables
SHELL	C	To increase environment size
SHIFT	B	To shift parameters
SORT	E	To filter and sort text lines
SUBST	E	To substitute a drive for a directory
SYMBOLS	-	Signs and symbols used by DOS
SYS	E	To transfer system (hidden) files
TIME	I	To display/change system time
TREE	E	To display directory structure

Command		Purpose
TYPE	I	To display a file
VER	I	To show DOS version
VERIFY	I	To verify writing after copying file(s)
VOL	I	To show a disk's volume label
XCOPY	E	To copy source files to memory and then to target

ANSI.SYS external

A special device driver file to enhance input (keyboard) and output (monitor) functions. This file must be included in your CONFIG.SYS before you can do the following.

Examples:

```
A>prompt $e[7m                  Set reverse video
A>prompt $e[2J                  Erase entire screen
A>prompt $e[0;68;"dir/p"p       Define F10 as dir/p
echo ^[[5;31;43m                Set blinking red text, yellow background
echo ^[[0m                      Turn off all color attributes
echo ^[[0;133;"cls$_"p          Define F11 as cls with carriage return
```

Remarks:

ANSI escape sequences (commands) are mostly entered via PROMPT with $e [and ECHO with ^ [[. The former is entered on the command line, and the latter can be entered only from a batch file created with EDLIN or another text editor. To define the extra keys on an enhanced keyboard, you must have v.4 and install ANSI.SYS with the /X switch. See Chapter 12 for details.

APPEND external, TSR

Make DOS search for nonexecutable (data) files through a path list.

Syntax:

```
A>append list
```

Switches:

/X or /X:ON	Search for executable files also
/X:OFF	Search only nonexecutable files
/PATH:ON	Search list with or without directory prefix from keyboard
/PATH:OFF	Search list only when no directory prefix from keyboard
/E	Store list in DOS environment, can be changed by SET
;	Clear path list

Examples:

```
C:\DOS>append /e          Enter into environment
A>append c:\xyz/x         Search for both exe and nonexe files in
                            C:\XYZ
A>append;                 Clear path list
```

Remarks:

APPEND stays in RAM after running. It could collide with other TRS (terminate and stay resident) programs and crash your computer. You can install it (use it the first time) with /E to put it in the environment and maneuver with SET. See Chapter 5 for APPEND and 9 for SET.

ASSIGN, JOIN, SUBST all external

Assign one drive to another; join a drive to an empty directory; substitute a drive for a directory.

Examples:

```
A>assign a=c b=c          Assign A & B to C; A & B inaccessible
A>assign                  Cancel any assignment
A>join b: c:\none         Join B to empty directory named NONE
A>join b: /d              Delete joining
A>join                    Show joined drives and directories
A>subst z: c:\wp\wk       Substitute Z (logical drive) for directory
A>subst                   Show list of substitution
A>subst z: /d             Delete substitution
```

Remarks:

Don't combine these three commands; you could get confused. Don't use them with: BACKUP, DISKCOPY, DISKCOMP, FDISK, FORMAT, PRINT, RESTORE. See Chapter 9 for more details.

ATTRIB external

Set or remove file attribute of A and R.

Syntax:

```
A>attrib +r +a filename
```

Switches:

+R	Set read-only
-R	Remove read-only
+A	Set archive
-A	Remove archive
/S	Subdirectories

Examples:

`C>attrib +r file? /s`	Set R for all matching files in current and subdirectory
`A>attrib +a +r file?`	Set A and R for all matching files
`A>attrib -a b:*.*`	Remove A attribute for all files in B
`A>attrib *.* ¦ sort`	List all files, sorted & with attributes
`C>attrib /s *.* ¦ find "FILE"`	List files matching `FILE`

Remarks:

A new or modified file automatically carries the A attribute. Some commands such as `BACKUP` and `XCOPY` can remove it after a file is copied. The R attribute prevents deletion, modification, and overwrite. Details in Chapter 7.

`BACKUP` external

Back up large files.

Syntax:

```
A>backup source target
```

Switches:

`/A`	Add to (not erase) files already on backup disk
`/D:date`	Date on or after which to back up
`/F:size`	Format target by specified size
`/L:name`	Log file name containing backup history
`/M`	Modified files only; changed since last backup
`/S`	Subdirectories are backed up
`/T:time`	Time at or after which to back up

Examples:

`C:\>backup wp\work a: /s`	Back up all files in `C:\WP\WORK` to A
`C:\WP\WORK>backup fil? a: /m/a`	Add modified files to backup file

Remarks:

`/A` will add to an existing backup file. All others will create a new backup file, erasing an old one if it exists. The root directory of target will be erased. Version 4 always needs access to `FORMAT.COM` before doing any backup. Use `RESTORE` to completely or partially restore files. See details in Chapter 7.

`BATCH` user-created

A batch file is an executable file that can contain a series of DOS commands, or be programmed to behave like a computer program. It must have the `BAT` extension in the file name. It can be created with `COPY CON` or in EDLIN. The following

commands, all internal, can be used in a batch file to make it behave like a computer program:

@	hide a line during execution
:	Mark a label
%	Replaceable parameter (dummy variable)
==	Test existence of parameters
CALL	Execute another batch file and return
ECHO	Turn on/off display; send data to a device
FOR..IN..DO	Repeat execution of a command
GOTO	Go to a label
IF (IF NOT)	Decision making for conditional branching
EXIST	Check existence of a file
PAUSE	Halt execution
REM	Remark
SHIFT	Shift to the next parameter

Remarks:

See ECHO and FOR, which are often used on the command line. See IF for an example of using @, :, %, ==, ECHO, GOTO, IF, EXIST, REM, and SHIFT. Chapter 11 provides more details of programming batch files and how to use each batch command.

BREAK see CONFIG.SYS

BUFFERS see CONFIG.SYS

CALL see BATCH

CD (CHDIR), MD (MKDIR), RD (RMDIR) all internal

Change directory; make directory; remove directory

Examples:

A>md b:temp	Make directory named TEMP in drive B
A>cd b:temp	Make TEMP the current directory in drive B
C:\WP>md misc	Make subdirectory named MISC, below C:\WP
C:\WP>cd misc	Change directory to MISC
A>rd b:temp	Remove TEMP directory in B, must be empty
C:\WP\MISC>cd\	Change to root directory, namely C:\
C:\WP\MISC>cd..	Change to parent directory, namely C:\WP
C:\WP>del misc	Delete all files in MISC subdirectory
A>cd	Display the path to the current directory

Remarks:

A directory must be empty before it can be removed. You can only remove a directory lower than the current one. Use `PROMPT $P$G` to show a default drive and current directory prompt. See `PROMPT` and Chapter 5 for more details.

CHKDSK external

Check a disk's formatted size, number of files and their total size, available disk space, allocation units, and available memory.

Switches:

`/F`	Fix errors
`/V`	Verbose; list files and their directory paths

Examples:

`A>chkdsk b:`	Check the entire disk in B
`A>chkdsk b:file?`	Check matching files' contiguity; file names displayed if not contiguous
`A>chkdsk b:*.*`	Check contiguity of all files
`C>chkdsk /v ¦ sort`	List all files in drive C, sorted
`C>chkdsk /v ¦ find "FILE"`	List files matching `FILE`

Remarks:

V.4 also shows volume label (if any), serial number (if formatted with v.4), and allocation units (clusters). If files are messed up, `/F` will create files named `FILEnnnn.CHK`, where nnnn is a number starting from 0000 and increasing as needed. See Chapter 3 for more details.

CLS internal

Clear the screen and place the cursor at the top left corner. See Chapter 1.

Syntax:

```
A>cls
```

COMP or FC both external

Compare binary or ASCII files.

Switches: `COMP` has no switch; `FC` has these:

`/A`	Abbreviated version, beginning and ending lines only
`/B`	Binary; compare byte by byte, show relative address

/C	Case ignored, all characters treated as uppercase
/L	ASCII; default for text files
/LBnn	Set internal line buffer to n lines, default 100 lines
/N	Display line numbers in ASCII comparison
/T	Tab not expanded to spaces
/W	Compress white spaces; tab as single space
/nnnn	# of mismatched lines before reporting difference

Examples:

```
A>comp *.txt *.bak          Compare all matching files
A>fc /a file1 file2         Abbreviated version
```

Remarks:

COMP comes with IBM and FC with other Microsoft releases. COMP will stop if the two files to be compared are of unequal lengths; FC will not. If differences are detected, COMP displays locations and ASCII values of different characters, all in hex numbers. When ASCII files are involved, FC will display more readable text lines. Details in Chapter 6.

CONFIG.SYS user-created

CONFIG.SYS is a special file to define your system. It can contain the following most commonly used configuration commands:

```
A>copy con config.sys
break on
buffers=20
files=20
device=ansi.sys /x
device=vdisk.sys
shell=c:\command.com /p /e:512
^Z
```

Remarks:

BREAK ON increases ^C checking; BREAK on or off can be entered on the command line. BUFFERS determines the number of disk buffers; FILES determines the number of files DOS can handle at one time. VDISK.SYS (or RAMDRIVE.SYS) lets you use a RAM disk. SHELL lets you expand the environment. See Chapter 8 for above. The /X switch of ANSI.SYS allows the use of an enhanced (101-key) keyboard; see ANSI for details.

COPY internal

Copy and combine files; copy files to devices PRN, CON, COM1, NUL.

Syntax:

```
A>copy source target      From source to target
A>copy source             Target is the default drive/directory
```

Switches:

```
/V      Verify target file
/A      ASCII, default for combining; read up to 1st ^Z and add ^Z to target
/B      Binary, default for copying; read entire source and add no ^Z to target
```

Examples:

```
A>copy *.* b:                Copy all from A to B, no file name change
C:\DOS>copy a:*.*            Copy all from A to current directory
A>copy f1+f2+f3              Append f2 and f3 to f1, ^Z added to f1
A>copy file?? file.all       Copy matching files to last, add ^Z
A>copy file+,,               Stamp current date-time on file, add ^Z
A>copy f1/b f2/a             Read entire f1, add ^Z to f2
A>copy/b file*.* b:+         Copy binary and stamp current date-time
A>copy/b prog*.* .+          Stamp current date-time on binary files
a>copy/b . .+                Stamp current date-time on all files
A>copy f1+f2 prn             Copy disk files to printer
```

Remarks:

Details in Chapter 6; see also COPY CON.

COPY CON

Copy to or from the screen; create or add to a file.

Examples:

```
A>copy con file1              (Create a file; copy from screen to file)
This creates a file named FILE1.
^Z
A>copy file1 + con            (Add to a file; add from screen to file)
This line is added to FILE1.
^Z
A>copy /b prog.exe con        Display file in binary format
A>copy file?? con             Display all the matching files
```

Remarks:

To end file creation, press F6 and Enter. The /B switch, which can be placed at the end of a command, uses the binary format to do copying. Without this switch, COPY CON will work the same as TYPE, stopping when the first ^Z is encountered. See also COPY.

DATE internal

Display and/or set date. See Chapter 1.

Syntax:

```
A>date
```

DEL internal

Delete named or matching files, which cannot be recovered except with software like Norton Utilities; it can't delete files marked R by ATTRIB.

Syntax:

```
A>del filename
```

Switches:

/P	Prompt for Y/N, available in v. 4.0 or later

Examples:

A>del fil	Delete FIL in drive A
A>del *.*	Delete all in drive A
A>del .	Same as above
A>echo y ¦ del .	Delete all, no prompt; dangerous to use
A>del b:fil*.txt	Delete matching files in B
A>del ??? /p	Delete files with 3 letters or less, prompt
C:\TEMP>del .	Delete all files in current directory
C:\>del misc	Delete matching file or entire subdirectory

Remarks:

It's dangerous to use DEL with wild cards. You can wipe out files you may intend to keep. Use /P switch when wild cards are used. In the last example, if MISC is a file, it is deleted immediately. If it's a subdirectory, you'll be prompted for Y/N before all the files are deleted. See Chapters 2, 4, and 5.

DIR internal

Display directory information.

Syntax:

```
A>dir filename/directory
```

Switches:

/P	Pause after a screen is full
/W	Wide format, only file names, not sizes or dates

Examples:

`A>dir`	Current directory of default drive
`A>dir file?`	Matching files
`A>dir b: /p`	B's current directory, one page at a time
`A>dir ¦ sort`	Sort by file name
`A>dir ¦ sort/+14`	Sort by file size
`C:\DOS>dir\`	C's root directory
`C:\WP>dir /w`	Current directory, wide format
`C:\BAT>dir..`	Parent directory
`C:\BAT>dir..\dos`	Show `C:\DOS`
`C:\>dir bat`	A single file or subdirectory
`C:\WP\REPT>dir ¦ find "SMITH"`	List all `SMITH` files

Remarks:

`DIR` cannot show hidden files. Use `CHKDSK` to detect them. See Chapters 2 and 5 for details.

`DISKCOMP` external

Compare two floppy, not hard, disks and report any discrepancies.

Syntax:

```
A>diskcomp disk1 disk2
```

Switches:

`/1`	Compare one side only
`/8`	Compare the first 8 sectors

Examples:

`A>diskcomp a: b:`	Compare disks in drive A and drive B
`C>\dos\diskcomp a: a:`	Load command from `C:\DOS`, 1-drive operation
`A>b:diskcomp a: b:`	Load command from drive B

Remarks:

If you have only one floppy drive, any of the above examples will work. You'll be prompted to swap disks. `DISKCOMP` is useful to compare the original and a copy produced with `DISKCOPY`. Comparison may not be meaningful if copy is produced by `COPY` or `XCOPY`. Details in Chapter 6.

`DISKCOPY` external

Duplicate disk's contents track by track. Target disk need not be formatted and its contents, if any, will be destroyed. Can be used to duplicate a system (bootable) disk

without formatting it with the /S switch. You can't diskcopy a hard disk. You'll be prompted to swap disks, if necessary.

Syntax:

```
A>diskcopy source target
```

Switches:

/1	Copy first side only

Examples:

`A>diskcopy a: b:`	Copy from A to B
`A>diskcopy`	Copy from A to A
`A>c:\dos\diskcopy b: a:`	Load command from C:\DOS, copy B to A

Remarks:

If no parameter is provided, one-drive copying will be performed. Files in target will be erased. Source should be write-protected. Don't diskcopy a fragmented diskette, use: `A>copy *.* b:`. See Chapter 6 for details.

ECHO internal and batch

Display ECHO state; echo text and control characters to devices.

Examples:

`A>echo`	Show ECHO on or off
`A>echo on`	Turn on ECHO
`A>echo off`	Turn off ECHO
`A>echo y ¦ del *.*`	Supply Y to delete all; use with caution
`A>echo print this line > prn`	Send one line to the printer
`xecution of batch lines`	
`echo/`	Create an empty line in a batch file screen display
`echo ^L > prn`	Send formfeed character to printer
`echo ^[[0;68;"dir/p"p`	Define F10 as dir/p in ANSI command
`echo Turn on the printer`	Echo message to the screen

Remarks:

ECHO is used mostly in batch files and ANSI escape sequences; see Chapters 8, 11, and 12 for more details of using this versatile command.

EDLIN external

Line editor. Must be loaded with a file name following it.

Commands:

.	Current line	M	Move lines
#	Last line	P	Page display
A	Append lines	Q	Quit, no save
C	Copy lines	R	Replace string
D	Delete	S	Search string
E	Exit and save	T	Merge text
I	Insert lines	W	Write lines
L	List lines	n	Enter n to edit line n
Enter	Press Enter alone to edit the line below current		
-/+	Before and after the current line		
?	Placed before R/S to show Y/N in search or replace		
^Z	Separate search and replace strings		

Remarks:

An old file is renamed with BAK extension. Control characters can be entered in EDLIN, not with COPY CON. ^Z is added at the end of each file. See Chapter 10 for details. See also KEYS for using editing keys.

ERASE internal

Same as DEL.

EXIST see BATCH and IF

FC see COMP

FDISK external

Partition a hard (fixed) disk so that DOS can use it to store data.

Syntax:

```
A>fdisk
```

Remarks:

A hard disk must be partitioned before it can be used by DOS. Version 3.3 and before can create a partition up to 32MB capacity; version 4 can go much higher. After the program is loaded, you will be prompted with a series of screens. You have numerous options to partition your hard disk. This needs to be done only on a new disk or when a hard disk crashes. Using `FDISK` will destroy any data on a hard disk. See Chapter 3 for more details.

`FILES` see `CONFIG.SYS`

`FIND` external

A filter command used to search for ASCII text lines. Can be used with `ATTRIB`, `CHKDSK`, `DIR`, `MORE`, `SORT`, and `TYPE` to find text lines or file names.

Syntax:

```
A>find "string" filename
```

Switches:

`/V`	Void; show lines not containing search string
`/C`	Count number of lines found
`/N`	Number; a number precedes each found line

Examples:

```
A>find "string" file      Display lines in FILE containing string
A>find /c "one" file      Count number of lines containing one
A>find /n "two" file      Display found lines with numbers
A>dir ¦ find /v "<"       Display directory without subdirectories
A>dir ¦ find "SMITH"      Find SMITH files and directories
A>find "no" file ¦ find "no"  Suppress display of file name
A>echo ¦ more ¦ time ¦ find "C" >> file
                          Append current time to FILE
A>for %x in (*.*) do find "string" %x
                          Find string in every file
```

Remarks:

When `/C` is specified, it can negate other switches. Details in Chapter 8.

FOR..IN..DO internal and batch

A looping command to repeat a command a number of times.

Syntax:

```
A>FOR %variable IN (file-list) DO command %variable
```

Examples:

```
A>for %a in (c:\wp\*.doc c:\ss\*.wk1) do copy %a b:
```
Copy all the matching files to B

```
A>for %x in (fil?) do type %x
```
TYPE all the files matching FIL?

```
A>for %m in (*.*) do del %m
```
Delete all the files, no warning

Remarks:

FOR can make a command do lots of work. You place a character (any character) after % as a variable. You then list in parentheses all the files you want to work on. Finally, specify a command and the variable. Some DOS commands, e.g., TYPE, do not accept wild card characters. The FOR command can bypass that limitation, such as the 2nd example. FOR is used most often in batch files; in that case, you must use %% before a variable. See Chapters 8 and 11 for details related to this command.

FORMAT external

Prepare floppy and hard disks for booting and file storage.

Syntax:

```
A>format b:
```

Switches:

/1	1 side only
/4	Format 360K in a 1.2MB drive
/8	8 sectors per track
/B	Blank space for later transfer of system files
/F:n	Disk capacity; v.4 only n = 160, 180, 320, 360, 720, 1.2 (or 1200), 1.44 (or 1440)
/N:n	Number of sectors per track, used with /T
/S	System files transferred
/T:n	Tracks per side, used with /N
/V:l	Volume label; if omitted, you'll be prompted in v.4

Examples:

```
A>format b: /f:360 /s
```
Format B, 360K in 1.2MB drive, system

```
A>format b: /n:9 /t:80
```
720KB in a 1.44MB drive

Remarks:

If no parameters are specified, a disk is formatted according to the drive's default capacity, such as 1.2MB. Formatting destroys any existing data. Details in Chapter 3.

GOTO see BATCH and IF

GRAPHICS external

Load GRAPHICS.COM; the program stays in the memory after that. You can then press Shift-PrtSc to print a graphics display. If the program is not in RAM, a graphics display will not be printed. See Chapter 2.

Syntax:

```
A>graphics
```

IF, ECHO, EXIST, GOTO, REM, SHIFT, @, %, :, == (all internal batch commands)

IF is used to test the existence of a parameter (line 4), of a file on disk (line 5), and error levels available with BACKUP, FORMAT, and other commands. GOTO branches batch execution to a label marked by :. SHIFT shifts to the next parameter entered on the command line; @ hides batch execution.

```
          1: @echo off
          2: rem TESTFILE.BAT--test existence of files
          3: :LOOP
          4: if .==.%1 goto END
          5: if exist %1 echo %1 exists
          6: shift
          7: goto LOOP
          8: :END
A>testfile jim jane jack
jim exists
jane exists
A>_
```

Remarks:

When this batch file is run, all the files placed after the batch file name will be tested. If a file is found, a proper message is shown; if not, nothing is shown. This batch file is created with EDLIN. If you use COPY CON to create it, take out the numbers and the colons right after them.

JOIN see ASSIGN

KEYS—Commonly Used DOS Keystrokes

F1	Copy from template 1 character at a time, same as Right Arrow
F2	Copy BEFORE 1st typed character; press F2, then a character
F3	Copy all remaining characters (from cursor to end)
F4	Copy FROM the 1st typed character; end with F3
F5	Save line to template; command is not executed
F6	Insert ^Z (end of file) character
F7	Insert ^@ (null, ASCII 0) character
Esc	Abandon edited line
Backspace	Delete 1 character to the left, same as Left Arrow
Del	Skip 1 character in template
Ins	Insert characters before copying from template
Ctrl-C	Terminate an action
Ctrl-P	Echo every move from console to printer
Ctrl-S	Halt screen display, same as Ctrl-Num Lock
Shift-PrtSc	Print screen display
Alt-Ctrl-Del	Reboot

Remarks:

The template is a memory area storing the most recently entered command. See Chapter 2 for the last five keys and Chapters 4 and 10 for the rest.

LABEL external

Display and/or change a disk's volume label.

Syntax:

```
A>label drive
```

Examples:

```
A>label b:          Show/change drive B's volume label
A>label disk2       Change drive A's volume label to DISK2
```

Remarks:

Version 4 will also display the serial number of a disk, which cannot be changed. See Chapter 3 for more details.

LASTDRIVE configuration

To designate the highest drive letter DOS can handle. Up to Z can be specified. If no LASTDRIVE is specified, DOS can handle only up to drive E. This is covered

in Chapter 9. To let DOS use all the possible drive letters, put this in your CONFIG.SYS file:

```
lastdrive = z
```

MD (MKDIR) see CD

MEM external

Show memory use. Available in version 4.

Switches:

```
/DEBUG
/PROGRAM
```

Examples:

`A>mem`	Show total and available memory
`A>mem /program`	Show above, plus programs already in memory
`A>mem /debug`	Show above, plus input-output devices

Remarks:

MEM, without any switch, shows memory use just like CHKDSK. The two switches will show amount of memory used by each program and their beginning and ending addresses. See Chapter 9 for more details.

MODE external, partially TSR

Set or report devices.

Syntax:

```
A>mode
```

Examples:

`A>mode con rate=32 delay=1`	Set highest keyboard speed, if supported by BIOS
`A>mode co40`	Set color, 40-column mode for CGA
`A>mode bw80`	80-column black and white for CGA
`A>mode lpt1 132, 8`	To print 132 columns and 8 lines per inch
`A>mode lpt1 80, 6`	Restore the default mode
`A>mode lpt1=com1`	Reroute printing to serial port
`A>mode con`	Report console status
`A>mode`	Report status of all devices

Remarks:

If you have EGA or VGA, you can display 43 lines on the screen. You need to specify the `/L` switch when installing `ANSI.SYS` (see ANSI command), and then enter this on the command line: `A>mode con lines=43`. See Chapter 9.

MORE external

A filter command to output ASCII text one screen at a time. Can be used with `ATTRIB`, `CHKDSK`, `DIR`, `FIND`, `SORT`, and `TYPE`.

Syntax:

```
A>more < filename
```

Examples:

```
A>more < filename                    Display ASCII text one screen at a time
A>type fil ¦ more                    Same as above
A>more > prn                         Print each text line entered on the screen.
A>dir ¦ more                         Display directory one screen at a time, like DIR/P
A>type fil ¦ find "Smith" ¦ sort ¦ more
                                     Show Smith lines, sorted, a screen at a time
A>echo ¦ more ¦ date ¦ find "C" >> file
                                     Append current date to FILE
```

Remarks:

`MORE` is useful in sending each line typed on the screen to the printer, like the 3rd example. The last example can be put in a batch file to insert the current date in a file. See Chapter 8 for details.

PATH internal

Make DOS search a list of paths for executable files. After a path list is entered, you can run an external command in any drive/directory without specifying a path prefix. DOS will go through the list to search for and execute the command.

Syntax:

```
A>path list
```

Examples:

```
A>path a:\;b:\;c:\bat                Search A:\, B:\, C:\DOS
A>menu                               Go to above list to execute MENU.BAT
C:\>path=c:\dos;c:\bat               Search C:\DOS and C:\BAT
C:\>tree                             Go to above list to execute TREE.COM
A>path                               List existing paths
A>path;                              Clear path list
```

Remarks:

PATH guides DOS to search for files ending with COM, EXE, or BAT extensions. Others are ignored. For these nonexecutable files, use APPEND to search for them. Use SET to see the current path list in the environment. See Chapter 5 for details.

PAUSE see BATCH

PRINT external, partially TSR

Print ASCII files in the background.

Syntax:

```
A>print file?
```

Switches:

/B:size	Buffer size in bytes, default 512
/D:dev	Device, such as COM1, default PRN
/M:max	Maximum number of clock ticks, 1-255, default 2
/Q:num	Queue; maximum number of files put in print queue
/S:slice	Time slices, 1-255, default 8
/U:wait	Wait clock ticks, 1-255, default 1
/C	Cancel printing files
/P	Print files
/T	Terminate all printing

Examples:

A>print /t	Terminate all printing; clear print queue
A>print f1/c f2	Cancel both files
A>print f1/c f2/p	Cancel F1, print F2
A>print	Show print queue

Remarks:

/C, /P, /T can be specified any time; others only the first time. See Chapter 7 for details.

PROMPT internal

Change the DOS prompt with text or metastrings shown below.

Switches (Metastrings):

e	Esc	l	Less than sign (<)
p	Current dir, default drive	b	Bar (¦)
g	Greater than sign (>)	q	Equal sign (=)
n	Default drive	h	Backspace

d	Date	-	New line
t	Time	$	Dollar sign
v	Version of DOS		

Examples:

`C>prompt $p$g`	Display current directory, default drive, and >
`A>prompt $n$g`	Default, showing A>
`A>prompt $t$g`	Display time and >

Remarks:

Can be used with `ANSI.SYS` to maneuver cursor position and screen colors. See ANSI and Chapter 5 for more details.

RD (RMDIR) see CD

RECOVER external

Recover file(s) from unreadable bad sectors and mark off bad sectors to prevent future use.

Syntax:

```
A>recover drive/filename
```

Examples:

`A>recover b:`	Recover all the files in drive B
`A>recover filename`	Recover a single file in drive A

Remarks:

Don't use `RECOVER` on an entire disk unless you have no other alternative; it will destroy the disk's directory structure. See Chapter 3.

REM see BATCH and IF

REN internal

Rename a file; give a new name to an existing file.

Syntax:

```
A>ren oldname newname
```

Examples:

`A>ren file.txt file.doc`	Change `FILE.TXT` to `FILE.DOC`
`A>ren file.txt *.doc`	Shorthand of above

```
A>ren b:file?.txt file?.doc
```
Change all matching files in drive B

Remarks:

A new file name cannot be the same as that of another file in the same directory. See Chapter 2.

REPLACE external

Replace old files with new ones of the same names. Read the syntax below as "putting source files in target."

Syntax:

```
A>replace source target
```

Switches:

/A	Add files that are in source but not in target
/P	Prompt for Y/N to replace old files
/R	Read-only files are also replaced
/S	Subdirectories in target; files there replaced
/U	Update old files with new (v. 4 only)
/W	Wait for a key press before acting

Examples:

`A>replace b:*.*`	Replace all in A with those in B if matching
`A>replace *.* c:/s`	Replace in C, incl. subdirectory, with those from A
`A>replace file? b:/a`	Replace B with A; add if not in target

Remarks:

Good for replacing old DOS files with new ones. /R can overwrite a source marked with R attribute; COPY and XCOPY can't. Details in Chapter 7.

RESTORE external

Restore file(s) from a backup file created by BACKUP.

Syntax:

```
C:\WP\WORK>restore a: file?
```

Switches:

/A:date	After; restore files backed up on or after date
/B:date	Before; restore files changed on or before date
/E:time	Earlier; files changed at or before specific time
/L:time	Later; files changed at or after specific time
/M	Modified; restore modified or deleted files
/N	None; restore files not in the target

/P	Prompt before replacing a changed file
/S	Subdirectory files are restored

Examples:

`C:\SS\BUDGET>restore a: budget1`	Restore a file to current directory
`C:\>restore b: file? /s`	Restore matching files to C and subdirectories

Remarks:

Must have a space after drive spec; must use original file names and directory, which was used as source by BACKUP. See Chapter 7.

SET internal

Display environment strings or enter environment variables.

Examples:

`A>set`	Display all the strings in the environment
`A>set var=one`	Put in the environment VAR as variable & ONE as its value
`set pa=%path%`	Assign existing PATH value to new variable PA
`set path=%pa%`	Assign PA to PATH
`set pr=%prompt%`	Assign existing PROMPT value to new variable PR

Remarks:

The DOS environment automatically contains PATH, PROMPT, and APPEND (if installed with /E); these can be entered without SET. You can use SET to enter your variables from the command line. Other uses such as the last three lines above must be done via batch files. See Chapters 9 and 11.

SHELL see CONFIG.SYS

SHIFT see BATCH and IF

SORT external

A filter command that can sort alphanumeric strings. Can be used with ATTRIB, CHKDSK, DIR, FIND, and TYPE to display output in sorted order.

Switches:

/R	Reverse (descending) order
/+c	Column number where c is the nth column from left margin

Examples:

```
A>sort < file /r          Sort FILE in reverse order
A>sort < file > file.srt  Save sorted file to a new file
A>sort < file /+10        Sort by the 10th column
A>dir ¦ sort              Sort directory by file name
A>dir ¦ sort ¦ more       Show sorted directory one screen
                            at a time
A>dir ¦ sort /+30         Sort directory by year
C:\>chkdsk /v ¦ find "SMITH"  List all SMITH files and directories
```

Remarks:

To sort items entered from the keyboard, run SORT without parameter, enter items, and press F6 and Enter to sort. SORT can't accurately sort numerals of unequal lengths. See Chapter 8 for details.

SUBST see ASSIGN

Symbols Symbols used by DOS for special purposes

<	Redirect input	==	Test parameter existence
>	Redirect output	+	Combine files
>>	Append to a device	*	Match remaining
¦	Pipe DOS commands	$	Metastring
.	Current directory	$e[	ANSI command (also ^[[)
..	Parent directory	@	Hide batch command
\	Root directory	:	Batch label
/	Switch	%	Replaceable parameter
?	Match 1 character	%x%	Environment variable
=	Assignment	^Z	End of file

Examples:

```
A>dir > file          Create FILE to store directory entries
A>dir >> file         Append to FILE
A>type file > prn     Redirect output to a printer
A>dir ¦ more          Pipe DIR output to MORE to display a screen
A>sort < file         Redirect FILE as an input file to be sorted
A>del .               Delete all in current directory
A>del file?           Delete matching files
C:\WP\X>cd..          Move up directory one level
```

Remarks:

See Chapter 2 for /; Chapter 4 for ? and *; Chapter 5 for ., . . ., \ and $; Chapter 6 for + and ^Z; Chapter 8 for <, >, >>, and ¦ ; Chapter 9 for = and %x%; Chapter

11 for @, :, %, %%, and == ; Chapter 12 for $e[and ^[[. EDLIN uses a separate set of symbols; see EDLIN and Chapter 10 for details.

SYS external

Transfer system (hidden) files.

Examples:

```
A>sys c:          Transfer hidden system files from A to C
```

Remarks:

The SYS command is used mostly to transfer a new DOS version's hidden system files from a floppy to a hard disk. SYS does not copy COMMAND.COM. To boot the new DOS version from the hard disk, you need to also use COPY to copy the new COMMAND.COM to the hard disk. See Chapter 8 for more details.

TIME internal

Display and/or set time. Version 4 will accept A (a.m.) and P (p.m.); earlier versions will accept only the 24-hour format.

Examples:

```
A>time
```

TREE external

Display directory structure. V.3.3 or earlier lists directory and file names. V.4 displays easier-to-read diagram.

Syntax:

```
A>tree drive                Version 3.3 or earlier
A>tree drive/directory      Version 4 only
```

Switches:

```
/F          List files, in addition to directories
/A          Use ASCII characters, instead of line draw, to display diagram; available
              in v.4 for speedier screen display
```

Examples:

```
A>tree b:                   Display directories in B
A>tree /f                   List files, plus directory, in A
C:\DOS>tree c:\misc         Analyze C:\MISC directory and subdirectory, v.4
```

Remarks:

V.4 can analyze a directory or drive. Others can handle only a drive. See Chapter 5 for details.

TYPE internal

Display a file's contents.

Examples:

```
A>type filename                          Display on the screen
A>type filename > prn                    Redirect to the printer
A>type fil ¦ sort ¦ more                 Show text, sorted, one
                                           screen at a time
A>for %x in (file?) do type %x           Show all files matching
                                           FILE?
```

Remarks:

TYPE works in ASCII mode. It stops reading when a ^Z (end-of-file) character is encountered. If you want to display a binary file on the screen, use COPY CON. TYPE can work with only one file at a time; no wild card character is allowed. You can bypass that by using the FOR command as shown in the last example. See Chapter 2 for details.

VER internal

Show DOS version.

Syntax:

```
A>ver
```

VERIFY internal

Display and turn on/off Verify.

Examples:

```
A>verify on          Turn on Verify
A>verify off         Turn off Verify
A>verify             To see VERIFY ON/OFF state
```

Remarks:

Verify is off by default. When turned on, writing to disk is verified. It has the same effect as using the /V switch with COPY. See Chapter 6.

VOL internal

Show volume label of a disk; version 4 also shows a disk's serial number. Change a volume label with the LABEL command.

Examples:

`A>vol`	Show drive A's volume label

XCOPY external

Read source file(s) to available memory, then copy to target.

Syntax:

```
A>xcopy source target
```

Switches:

/A	Archive, only marked files copied
/D:mm-dd-yy	Date, copy files on or after date
/E	Empty, copy empty directories, used with /S
/M	Remove archive bit in source after copying
/P	Prompt for Y/N before copying each file
/S	Subdirectory, copy subdir from source to target
/V	Verify target file
/W	Wait for a key press before copying

Examples:

`A>xcopy *.* b:`	Copy all from A to B, no file name change
`C:\DOS>xcopy *.* b:/s`	Copy C:\DOS and subdirectories to B
`A>xcopy file??.* /p`	Copy matching files, prompt Y/N
`A>xcopy *.* b: /a`	Copy all marked with A attribute

Remarks:

XCOPY is improved COPY. It works faster with large files; can handle subdirectories and back up dated files. See Chapter 6 for details.

Appendix E

Installing and Using the Shell

There are quite a few shell programs available for DOS; some of these are sold and others available from computer bulletin boards. A shell program provides a text or graphical environment to insulate users from the individual DOS commands. A user can point to a command displayed on the screen and press Enter or click a mouse to execute it. Help in using the command is usually available should you need it.

DOS version 4 can create a shell environment for you if you choose it during the installation process. A batch file named `DOSSHELL.BAT` will be created containing the information given by you during the installation. This batch file will execute `SHELLB.COM` and `SHELLC.EXE` (stored on the Shell disk) to create a graphical environment.

To use the DOS shell, you need to follow the installation process. This process is quite elaborate. If you do not want to use the shell, the process is unnecessary and a waste of time. If you just want to upgrade to version 4, there are much easier ways. See Chapter 7 for details.

When you boot DOS with the Install disk in drive A, the `AUTOEXEC.BAT` file stored on the disk runs the `SELECT.COM` program and gives this instruction:

```
Insert SELECT diskette in drive A:

Press Enter to continue installing DOS,
or press Esc to exit.
```

If you press Esc, DOS is booted and the system prompt of `A>` appears. You can then do whatever you want. If you press Enter instead, the disk drive will get busy for a while and then you will be asked to put the Install disk back. Then this message appears:

```
                              Welcome

   Welcome to DOS 4.0 and the SELECT program. SELECT
   will install DOS 4.0 on your fixed disk or diskette.
   If you install DOS 4.0 on a diskette, the number of
   blank diskettes you need depends on the type and
   capacity of your diskette drive:

      Drive Type (Capacity)        Number of Diskettes

      5.25-Inch Drive (360KB)      five  5.25 (360KB)
      5.25-Inch Drive (1.2MB)      five  5.25 (360KB)
      3.5-Inch Drive (720KB)       three 3.5  (720KB)
      3.5-Inch Drive (1.44MB)      three 3.5  (720KB)

   If you install DOS 4.0 onto a fixed disk, you need
   one blank diskette:

      5.25-Inch Drive              one   5.25 (360KB)
      3.5-Inch Drive               one   3.5  (1 or 2MB)

      Press Enter (↵) to continue or Esc to Cancel

   Enter    Esc=Cancel
```

At this point (and in future screens), you are to press either Esc to end or Enter to continue. If you press Enter, another screen appears:

```
                            Introduction

   As you view the SELECT displays, you will be asked to make
   a choice or type an entry. If you are uncertain about what
   to choose or type, you can accept the predefined choice or
   press the F1 help key for more information about an item.

   You will be using these keys in the SELECT program:

      Enter              To proceed to the next step.
      Esc                To cancel the current display.
      Tab                To move to the next entry field.
      PgUp/PgDn          To scroll information one page at a time.
      Up/Down Arrow      To move the highlight bar to the next item.
      F1                 To view the help information.
      F3                 To exit SELECT.
      F9                 To view key assignments while viewing help.
      Left/Right Arrow   To scroll data fields horizontally
                           to the left or right.
```

```
You can press Enter, Esc, and F1 keys when they appear on the display.

Enter    Esc=Cancel
```

Pressing Enter displays another screen:

```
                    Specify Function and Workspace

    SELECT sets up your computer to run DOS and your programs
    most efficiently based on the option you choose.
    Note:  You can review the results of your choice later
    in this program.

    Choose an option:

        1. Minimum DOS function; maximum program workspace

        2. Balance DOS function with program workspace

        3. Maximum DOS function; minimum program workspace

Enter    Esc=Cancel    F1=Help
```

Several more screens will request you to enter information related to your hardware. Then you will be asked to swap disks so that DOS commands can be copied to the hard disk (if you choose to install on your hard disk). If your hard disk runs out of disk space, an error message is displayed and the installation (copying) is aborted. You need plenty of disk space to store all the DOS commands.

If the installation is successful, two more batch files are created, `AUTOEXEC.400` and `CONFIG.400`. These files contain suggested commands to operate your PC efficiently. Use the `TYPE` command to examine them. If you want to use them without modification, simply change their names to `AUTOEXEC.BAT` and `CONFIG.SYS`. If you wish, you could incorporate some lines into your existing files and erase these two files. EDLIN is a handy tool on such an occasion.

In the future, when you boot DOS from the disk containing these files, a window will automatically appear instead of the DOS prompt. The window will show a number of options. You can move to each option with a mouse and click, or with an arrow key and press Enter.

The initial shell screen looks like this:

```
06-22-90                    Start Programs                    10:35 am
Program  Group  Exit                                          F1=Help
-------------------------------------------------------------------------
                            Main Group
    To select an item, use the up and down arrows.
    To start a program or display a new group, press Enter.

Command Prompt
File System
DOS Utilities
```

Right under the date-time line, a number of options for action are shown with the hot key underlined; you can just press the underlined key to perform an action. This line changes as you select an item in the main part of the screen.

The middle part of the screen shows three menu entries. If you have a color system, a fourth option also appears to let you change screen colors.

The cursor rests on the first option, which takes you to the DOS prompt. You can enter whatever command you wish. When done, enter the `EXIT` command to return to the shell.

If you select File System, the screen is split into two halves, with the left showing the current directory's tree structure and the right a list of files similar to the screen produced by the `DIR` command. You can now select Options to view, move, print, open (execute), delete, rename files, and take other related actions.

If you select DOS Utilities, you will be given these options:

```
Set Date and Time
Disk Copy
Disk Compare
Backup Fixed Disk
Restore Fixed Disk
Format
```

These options simply let you execute external DOS commands without leaving the shell.

The shell provides plenty of hand-holding each step of the way. You can always press F1 to show a small screen with instructions. When you want to enter a DOS command, another small window opens up for you to type required parameters. If you run an application program, the window disappears and the new program takes over. When you exit the application, the shell interface reappears.

The shell stays in memory as you work with your PC. It takes up the amount of memory determined by your choice during the installation. This could deprive your application program of the memory it needs to operate efficiently.

The shell is useful for people who are casual users of DOS and have no desire to learn to master many cryptic commands. By using the graphical environment, you rarely come into direct contact with individual DOS commands; you don't even have to spell a command correctly. From one window to another, you can get to where you want without having to enter a DOS command.

Using the shell naturally means that you know little about DOS. If something goes wrong and you can no longer just point and shoot, you are not likely to be able

to handle it. Such occasions require you to know various DOS commands, switches, and parameters. You are likely to be ignorant of these if you use the shell regularly.

The DOS shell has not caught on. Those who are versed in DOS consider it a bother. If you want to order DOS to do something, why not just give a direct order? Some companies that do not want to burden their employees with complex DOS commands find it easier using other shell packages than the one provided with version 4.

When version 4.0 first came out, IBM and Microsoft touted the shell and the graphical environment it creates as a major improvement. They used this new addition to justify a hefty price increase over version 3.3. As this environment is not widely appreciated, version 4.01 is now sold with a skimpy premium over version 3.3; some vendors even sell them at the same price.

Appendix F

Elementary Computer Math

A computer may appear intelligent, but at its basic level it can "understand" only whether power is on or off. To give it an instruction, we need to go by 1 (on) or 0 (off). The two digits of 1 and 0 form the basis of binary (base 2) math.

The building block of binary math is the bit, which is a contraction of two words—binary digit. A bit can have the value of either 1 or 0.

When you read binary numbers, read from right to left. When a digit goes beyond 1, the excess is carried to the next number on the left. Consider the following examples of equivalent binary and decimal (base 10) numbers, and the methods for converting the former to the latter:

Binary	Decimal	Calculation
1	1	1
10	2	2+0
11	3	2+1
100	4	4+0+0
101	5	4+0+1
110	6	4+2+0
111	7	4+2+1
1000	8	8+0+0+0
1001	9	8+0+0+1
1010	10	8+0+2+0

Each bit has the value of 2 to a certain power. The first (rightmost) is 2 to the power of 0 (by definition, 2^0 = 1), the second 1, the third 2, etc., as shown below:

Decimal		128	64	32	16	8	4	2	1
2^n	n =	7	6	5	4	3	2	1	0

Based on the above table, 2 to the power of 7 is 128.

Depending on whether a bit is on or off, the following binary number:

00111001

can be translated to its decimal equivalent this way:

0 + 0 + 32 + 16 + 8 + 0 + 0 + 1 = 57

By putting 8 bits together, as in the above example, you will get a byte and up to 256 variations (11111111 in binary equals 255 in decimal). The 256 ASCII values, shown in Appendix B, are based on this principle. The following table shows some examples of decimal, binary, and ASCII equivalents:

Decimal	Binary	ASCII Character
49	00110001	1
50	00110010	2
51	00110011	3
...		
65	01000001	A
66	01000010	B
67	01000011	C
...		
97	01100001	a
98	01100010	b
99	01100011	c

Bytes can go up to thousands, millions, billions, and more. For simplicity, programmers use KB (kilobyte), MB (megabyte), and GB (gigabyte). These numbers coincide with some unique numbers as shown below:

Power of 2	Abbreviation	Decimal
2^10	KB	1024
2^20	MB	1048576
2^30	GB	1073741824

Thus, 1KB is 2 to the power of 10 (2 multiplied by itself 9 times or 2×2×2×2×2×2×2×2×2×2), 1MB the power of 20, and 1GB the power of 30.

Binary numbers are difficult to maneuver, so programmers use octal (base 8, 1 byte) and hexadecimal (base 16, 2 bytes). An octal digit can be 0 to 7. A hex digit can be any of the following 16:

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

These numbering systems belong to the domain of computer programmers and most of the time should not concern you.

Appendix G

Further Reading

(Note: Most of the following PC-related periodicals are available in many bookstores, computer stores, college libraries, and public libraries. If you don't find them there, contact the publishers. The computer press is highly volatile. Some of these publications may no longer exist by the time this book is published; new ones may appear.)

Periodicals

Academic Computing

This monthly publication (no publication for June, July, August, and December) contains mostly articles written by professors and administrators dealing with computer use in colleges and universities. Software vendors like to announce here their special discounts for college students. Previously free to subscribers, it began in 1990 to charge variable subscription fees of $30 to $75. Write to: *Academic Computing*, P.O. Box 840, McKinney, TX 75069; or call (214) 548-2101.

Byte

Published by McGraw-Hill, this monthly provides comprehensive coverage of small computers of all kinds. One of the few surviving publications that accompanied the onset of the PC revolution, it recently celebrated its 15th anniversary. *Byte* has adapted to the rapidly changing world of microcomputers. As the PC has become more popular, *Byte* has concentrated more on it, shifting from earlier emphasis on Apple, Atari, and Commodore. You can find tidbits about new products, reviews of

individual products, questions to the editors and their answers. The magazine operates BIX (Byte Information Exchange) online service for members who pay extra fees. You can also find frequent ads by various book clubs soliciting members; joining a book club is an inexpensive way to acquire the books you really need. To subscribe, contact the Subscription Department, P.O. Box 558, Highstown, NJ 08520-9409. Annual subscription: $24.95.

Home Office Computing

Previously known as *Home and Office Computing*, the skimpy (about 100 pages) monthly is primarily intended for people who use their homes as offices. Most articles provide advice for people who use PCs. Coverage of PC and Mac hardware and software items commonly used in the home office setting. Published by Scholastic, Inc., 730 Broadway, New York, NY 10003. Annual subscription: $19.97.

InfoWorld

This weekly resembles *PC Week* in contents and appearance. For a subscription, contact *InfoWorld*, P.O. Box 5997, Pasadena, CA 91117; or call (818) 577-7233. Annual cost is $100.

PC Computing

Aimed at individual and business users of the PC; there is occasional coverage of the Mac. Brief news about new or updated products, and half-page reviews of some recently released popular products. Half a dozen columns by PC veteran writers. There are tips and hints provided by experts and readers on various popular applications such as dBASE, Lotus 1-2-3, and WordPerfect. Every issue includes some help or hints for DOS. If you have tenacious bugs or problems with your hardware or software, you can write to them. If their staff cannot solve the problems, readers will be offered $50 to come up with solutions. Published once a month by Ziff-Davis. For subscription, write to PC Computing, P.O. Box 58229, Boulder, CO 80321-2770; or call (800) 365-2770. Annual subscription: $24.97 for 12 issues. The company also offers software giveaways to entice subscribers.

PC Magazine

Published twice a month, 22 issues for one year, by Ziff-Davis. This is a substantial (about 400 pages) and comprehensive magazine. It has typical new and updated software and hardware news, and columns by veteran PC writers. Each issue thoroughly tests and compares one or two kinds of products (software and hardware), such as laser printers. Those designated as *PC Magazine* Editor's Choice are frequently cited by vendors in their advertisements. You should consult these comparisons and test results before you buy an expensive item. Each issue contains some tutorial material related to DOS and other popular programs. If you have questions, you can ask the Tutor. If you know some neat tricks, you may be paid $50 for your contribution. For a subscription, contact *PC Magazine*, P.O. Box 54093, Boulder, CO 80322; or call (800) 289-0429. The annual subscription is $44.97. In a recent promotion, it was $29.97, plus a bonus of two utility diskettes. The company

also operates PC MAGNET online service, from which members can download existing programs.

PC Today

Billed as "the magazine for computer newcomers," this is a relatively new and skimpy (about 100 pages) monthly. It features some reviews of software packages for beginners. The February 1990 issue compares quite a few menu programs for DOS. The most useful feature is a comprehensive and continuously updated database containing a list of numerous PC products and the vendors with lowest advertised prices. This database is useful if you intend to buy something from a mail-order merchant. The prices listed here may not be completely accurate because products may not be comparable or equally equipped; some vendors may also impose other charges. For a subscription, contact Peed Corporation, 120 West Harvest Drive, Lincoln, NE 68502. Annual subscription: $19.

PC Week

Latest news about hardware and software products and companies in the PC industry, with emphasis on corporate use and development of PC products. Useful weekly Buyer's Guides comparing one kind of product, such as dot-matrix printers, with charts supplying detailed specifications for each. Opinions of those who have used or tested new or updated products. Several weekly columns written by veteran writers. Occasional coverage of non-PC products such as Macs and workstations. Published by Ziff-Davis. For a subscription, write to: Consumer Service Department, *PC Week*, P.O. Box 5970, Cherry Hill, NJ 08034; or call (609) 428-5000. Annual subscription: $160. Free subscription available to qualified persons and institutions.

PC World

This is a slightly smaller (about 250 pages) and less comprehensive version of *PC Magazine*. Published monthly by PCW Communications, Inc. For a subscription, contact Subscriber Service, P.O. Box 55029, Boulder, CO 80322-5029. In a recent promotion, the annual subscription was $19.97, plus a free calculator.

Personal Computing

A smaller version of *PC World*. Published by Hayden Publishing Company. Annual subscription: $18; discounted during promotion. Call (800) 525-0643 or write to *Personal Computing*, P.O. Box 2942, Boulder, CO 80322.

Other Publications

Other publications that are less general in nature include *Computer Language, Dr. Dobb's Journal, Microsoft Systems Journal, PC Tech Journal,* and *Programmer's Journal* for programmers; *Computer Shopper* and *Computer Buyer's Guide* for bargain hunters; *Personal Publishing* and *Publish!* for desktop publishing; *AI Expert, Computerworld,* and *Information Week* for computers in general. Many of these are available in college libraries.

Other Sources

Besides print media, a variety of electronic media can supply you PC-related news. These include TV shows, bulletin boards, and fee-based databases.

Some TV broadcasts are available on cable and, from time to time, on PBS. They provide relatively shallow coverage.

Most large cities have bulletin boards run by individuals, computer clubs, or public colleges. These contain lots of free (shareware or public domain) DOS games, graphics, applications, and utilities that can be downloaded to your disk by using a modem.

Information providers such as CompuServe, Source, and GEnie have special interest groups (SIGs) devoted to all kinds of specialties such as DOS, Lotus, WordPerfect, etc. You need a modem and pay various amounts of fees to access them; their subscription packages are readily available in computer stores and frequently advertised in computer publications.

The University of Washington maintains a nationwide system called ISAAC (Information System for Advanced Academic Computing). Qualified academicians are given a password and a disk containing Kermit (a communications program) for accessing the system via a toll-free (800) number. There is no fee (other than buying a modem) for using the system. Contact the following for details: ISAAC, m/s FC-06, University of Washington, Seattle, WA 98195.

Appendix H

Solutions

Chapter 1: PC and DOS

Drill

1.e; 3.b; 5.d; 7.c; 9.d; 11.e; 13.d; 15.a; 17.b; 19.c; 21.S; 23.S; 25.S; 27.T; 29.F.

Practice 1-1

1. Press the Reset button on the front panel of the system unit.
3. Type something and then press Esc. The cursor goes down one line where a new command can be entered.
5. Enter the `VER` command.
7. Enter the `TIME` command. When the prompt for new time appears, type the same hour, then a colon, and then the new minute. Press Enter to register the command.

Review Questions

1. PC (Personal Computer) is the term used by IBM when it first introduced its small computers. XT (Extended Technology) is an improved PC. AT (Advanced Technology) was used when IBM first introduced its 286-based PCs. PS/2 (Personal System/2) is the name IBM has given to its current crop of PCs.
3. If it's 8080 or 8086, it is the slowest and most limited; 80286 is in the middle range; 80386 is in the top range; 80486 is rare and very powerful.
5. CGA-RGB is the crudest color system. A Hercules monographics system is the most popular noncolor system. EGA is still widespread, but VGA is rapidly becoming the most popular graphics system. It can accommodate both mono-

chrome (analog) and color monitors to display high-resolution characters and graphics.

7. PC, AT (larger Enter key and some lighted panels) or Enhanced (12 function keys on top, extra cursor keys, etc.).
9. Restart your PC (warm boot), just like the Reset button available on newer PCs.
11. DOS for Intel lines of CPUs. It's a single-user, single-tasking system. OS/2 for Intel CPU of 80286 or higher. It is basically a multitasking system. UNIX, originated from AT&T's Bell Lab, is basically a multiuser system; it runs on 80386 or higher CPUs.
13. It does if date and time are kept up to date. It maintains power supply to the calendar and keeps it up to date even when power is off.
15. Simply enter the `DATE` or `TIME` command on the command line to change the numbers.

Chapter 2: A Quick Tour Through DOS

Drill

1.T; 3.T; 5.F; 7.b; 9.T; 11.T; 13.T; 15.d; 17.a; 19.d; 21.c; 23.a; 25.T; 27.F; 29.F.

Practices

2-1

1. `A>b:`
   `B>a:`
3. `A>dir b: /w`

2-2

1. `A>copy sample.txt b:myfile.doc`
3. `A>type b:file2.txt`

2-3

1. `B>ren myfile.doc newdoc`
3. `A>b:`
   `B>del file2.txt`

2-4

1. `A>dir` (and then press Shift-Print Scrn)
3. `A>type sample.txt` (do not press Enter)
   Press Ctrl-P.
   Press Enter.
5. `A>copy sample.txt prn`

Review Questions

1. A collection of related data. `COM` is a command file, `EXE` is an executable file, and `BAT` is a batch file. DOS can run all three of them. `SYS` is a device driver that can define or alter your system.

3. Where a command or file will automatically be accessed. To change it, simply enter a drive letter followed by a colon, such as `B:`.
5. `/P` to pause after a screen is full; `/W` to display wide format.
7. A switch is specific with each command. Some commands have no switch while others have more than half a dozen. A switch is placed after a backslash. A parameter is a command-line argument telling DOS how to do a job such as specifying a different target drive.
9. Yes, if you store different copies in separate directories. Otherwise, you must use a different name.
11. Will format it if it is not already formatted. Will destroy the contents, if any. Will put the source disk's contents on the target disk.
13. A echoes every keystroke to the printer. B halts a screen display. C prints a screen display.
15. Prints a file named `FILENAME`. Actually, a disk file is copied to the printer. The screen also shows the message `1 File(s) copied`.

Chapter 3: Managing Disks

Drill

1.b; 3.a; 5.F; 7.T; 9.F; 11.tracks; 13.512; 15.F; 17.c; 19.d; 21.T; 23.F; 25.T; 27.T; 29.F.

Practices

3-1

1. `A>vol`
3. `A>label b:`
   If a volume label is not shown: Enter a name.
   If a volume label is shown: Press Enter, N, Enter.

3-2

1. `A>chkdsk b:`
3. `A>chkdsk *.*`

Review Questions

1. CD-ROM stands for Compact Disk–Read Only Memory. It resembles an audio compact disk. Huge amounts of data can be stored on it. The data normally cannot be erased by the user. DAT stands for Digital Audio Tape. This is a relatively new storage media. It can store large amounts of data and also provide for slow random access. It is erasable.
3. A 1.2MB drive can format a disk to 360KB or 1.2MB, and can read and write to either. A 360KB drive can read only a 360KB, but not 1.2MB disk. A 1.44MB drive can format both 1.44MB and 720KB disks as well as read and write to them. A 720KB drive cannot handle a 1.44MB disk.

5. A disk is divided into a number of circular rings called tracks. Each track is divided into a number of sectors. Each sector has the storage capacity of 512 bytes.
7. The smallest unit DOS can use to save a file. A cluster can be one, two, or more sectors (512 bytes). When a file is saved, no matter how small, it takes up a whole cluster.
9. This is available with version 4.0 or later. It allows you to specify byte size to format a disk.
11. Volume label, serial number (v.4), number of files (including hidden), cluster size (v.4), available disk space and memory.
13. This command will show all the fragmented files:
    `A>chkdsk *.*`
15. Just like a series of floppy disks stacked together. Each disk is known as a platter, which is divided into a number of tracks known as cylinders.

Chapter 4: Creating and Naming Files

Drill

1.F; 3.c; 5.d; 7.T; 9.a; 11.F; 13.c; 15.h; 17.e; 19.d; 21.i; 23.a; 25.F; 27.a; 29.F.

Practices

4-1

1.
```
A>copy con quote
Live simply so others may simply live.
^Z
```
3.
```
A>ren quote quotes
```

4-2

1.
```
A>copy con quota
He who knows does not speak.
^Z
```
3. `A>del quot* /p` (no /p allowed before v.4)

4-3

1.
```
A>copy con filename
You can't be all things to all people.
```
3. Press F4, then e, then Ins, then B, and then F3.

4-4

1.
```
A>copy con f.bat
c:\dos\format  a:  (a:format b:)
^Z
```
3.
```
A>copy con ck.bat
a:chkdsk  %1:
^Z
```

Review Questions

1. You ask DOS to copy what you type from the console to the named file in the current directory. In effect, this creates a new file.
3. Press Ctrl-C or Ctrl-Break. A ^C character appears and the file is not saved to disk.
5. Space, period, comma, <, >.
7. No; ? is ignored and characters after * are discarded.
9. Yes for REN, no for TYPE. TYPE is directed at a specific file. It is not capable of displaying a group of files.
11. You abandon the command, and DOS will not execute it. The cursor goes down to the beginning of the next line for you to enter another command.
13. To give the user more flexibility. It can be replaced with any parameter entered from the keyboard. One batch file can thus be used to do a number of tasks, depending on the parameter entered.
14. In the root directory of the boot disk, the same place as COMMAND.COM.
15. Your computer will freeze.

Chapter 5: Managing Directories

Drill

1.F; 3.F; 5.d; 7.a; 9.b; 11.d; 13.d; 15.a; 17.e; 19.c; 21.F; 23.F; 25.T; 27.F; 29.T.

Practice

5-1

1.
```
A>md test.1
```
3.
```
A>rd test.1
A>dir
```

5-2

1.
```
A>md temp
```
3.
```
A>copy b:filename
```
5.
```
A>rd temp
```

5-3

1.
```
A>md alpha
A>md beta
A>md gamma
A>md delta
A>cd alpha
A>md a1
A>md a2
A>md a3
A>cd \delta
A>md d1
A>md d2
```

```
   A>md  d3
3. A>copy filename \delta\d3
5. A>dir\
```

5-4

```
1. A>prompt $p$$
3. A>prompt $t$h$h$h$h$h$h$g
```

5-5

```
1. A>tree b:
3. A>tree b:\delta
5. A>append /x
   A>append a:\delta
   A>append
```

Review Questions

1. Different persons can use the same disk without undue interference. Related files can be put in a directory for easier remembrance and access. Different software packages can coexist and not interfere with one another. It takes a shorter time to display available files in a directory. Being a separate unit, a directory can be managed more efficiently. Files of the same name can be stored on the same disk in different directories.
3. The active directory, where all actions can be taken without telling DOS where to apply them. Thus, when you use the command `DIR` alone, the current directory's files are displayed.
5. This command will do:
   `A:\SPORTS\SHOPS>md\runner`
   The `\` sign tells DOS to create a directory just below the root directory.
7. Use the `CD..` command.
9. The screen displays the default prompt, which includes the default drive letter followed by the greater than (>) sign.
11. `PROMPT $D$_$N$G`
13. You ask DOS to go to `C:\DOS` to run the `TREE` program. In version 3.3, the screen shows a list of directory names of the entire disk in the default drive (A). Version 4 displays the tree structure (diagram) of the current directory and those below.
15. To display on the screen the previously entered path specifications. To clear a path list, put a semicolon (`;`) after `PATH` or `APPEND`.

Chapter 6: Copying and Comparing Files

Drill

1.b; 3.b; 5.d; 7.d; 9.c; 11.d; 13.d; 15.F; 17.F; 19.T; 21.F; 23.d; 25.E; 27.I; 29.E.

Practices

6-1

1. A>copy con test1
   This is test1.
   ^Z
   A>copy con test2
   This is test2.
   ^Z
3. A>copy test3 + test1 + test2
5. A>copy test? . +

6-2

1. A>copy /b test1 con (change 1 for other files)
3. A>copy test2 + test3 prn

6-3

1. A>md testdir
   A>cd testdir
   A>copy test1 testdir
   A>del test1
3. A>b:
   B>dir
   A>dir testdir

6-4

1. A>diskcomp a: b:
3. A>del test?
   A>del testdir
   A>rd testdir
   (repeat above for drive B)

Review Questions

1. Copies all the files with DOC extensions from drive B to drive A and changes the extensions to BAK. This is a binary operation, and no ^Z is added to target files.
3. Appends all the files in the current directory of drive B to the named file. An extra ^Z is added to the target file.
5. Binary mode for copying. Everything in the source is read and written to the target. In file combination, ASCII mode is the default. A source file is read until a ^Z is encountered. A target file has an additional ^Z added. The /A and /B switches can be used to counter these default settings.
7. The first copies each of the files from drive A to B. The fragmented pieces of each file are put together before writing to drive B. The target disk needs to be formatted and has enough space to hold the incoming files. The second physically duplicates in drive B the disk in drive A. A fragmented original produces an equally fragmented copy. The target disk need not be formatted. A bootable disk can also be duplicated this way.
9. A>copy filename prn
11. Duplicates in the target drive the directory structure of the source. If target directories do not exist, they are created before copying files.

13. No to both.
15. COMP yes, FC no.

Chapter 7: Maintaining Files

Drill

1.F; 3.T; 5.T; 7.a; 9.F; 11.F; 13.T; 15.c; 17.c; 19.T; 21.T; 23.a; 25.d; 27.d; 29.T.

Practices

7-1

1. A>copy con test1
   This is test1.
   ^Z
   A>copy con test2
   Here is test2.
   ^Z
   A>replace test? b:
3. A>copy test1 + con
   This line is added.
   ^Z
   A>replace test? b: /u

7-2

1. A>attrib test?
3. A>replace b:test? /r
5. A>attrib -r test?
   A>del test?
   A>replace b:test?

7-3

1. A>backup test? b: /l
3. A>type b:backup.001

7-4

1. A>del test1
   A>restore b: test1

7-5

1. A>print test?
3. A>print test1 test2

Review Questions

1. To transfer hidden system files of IBMBIO.COM (IO.SYS), IBMDOS.COM (MSDOS.SYS).
3. Add files from source to target. If matching files are found in B (source) but not in A (target), they are added to A.

5. All the files in A are set to read-only attribute.
7. Access will be denied. XCOPY cannot overwrite a file with the R attribute.
9. To create sequentially numbered backup files to store original files. Two files are created by BACKUP, namely BACKUP.nnn and CONTROL.nnn. The former contains the contents of the original files and the latter the control mechanism to restore them.
11. Back up every file in C:\WP to drive A. A new backup file will be created and the files in the root directory of A will be erased.
13. Each time BACKUP backs up a file, it strips off the file's A attribute. When you use it with the /M switch, it will skip those without the A attribute. That means only modified or newly created files are included.
15. Cancel the first two files and print the last.

Chapter 8: Redirecting Files

Drill

1.d; 3.b; 5.a; 7.d; 9.a; 11.F; 13.F; 15.F; 17.F; 19.F; 21.T; 23.T; 25.T; 27.T; 29.c.

Practices

8-1

1.
```
A>copy con xyz
Line 1.
This is line 2.
Here goes line 3.
^Z
```
3.
```
A>type xyz > nul
A>type xyz > prn
```
5.
```
A>copy con cr

^Z
A>copy con t.bat
time < cr
^Z
```

8-2

1.
```
A>more > prn
line 1.
line 2.
line 3.
^C
```
3.
```
A>more xyz > prn
```

8-3

1. `A>copy con address`
   `Drake, Daffy  123 Kwack Ave., Fun City  CA  94000`
   `Bunny, Bugs  999 Doc Road, Mud Town  OK  74037`
   `. . .`
   `^Z`
3. `A>sort < address /r`
5. `A>sort < address /+43 >prn`
   (count the longer line, including space and tab)

8-4

1. `A>find "CA" address`
3. `A>find "CA" address ¦ find "Smith" address`
5. `A>copy con seeadd.bat`
   `find "%1" address`
   `^Z`

8-5

1. Change all the `diary` occurrences in the text to `note` or `notes`. Also replace the `MORE` lines in the text with those in `DATEFILE.BAT` in Tips & Tricks.
3. `A>seeline notes Smith`

Review Questions

1. Keyboard is standard input, and monitor standard output. The former is used to enter raw data, and the latter to output processed data.
3. 1. Saves directory entries to `FILENAME`.
   2. If existent, `FILENAME` ignored; if not, command aborted.
   3. Appends to `FILENAME`.
5. 1. Line is echoed to `NUL`, not duplicated on the screen.
   2. Line is printed.
   3. Line is saved.
   4. Line is appended.
7. `MORE` is run without any parameter. After exiting `MORE` by entering `^Z`, `DIR` is executed.
9. Those of equal lengths. `SORT` treats each character as an ASCII value, except adjusting for uppercase and lowercase. Numbers and letters are sorted according to their ASCII values, not numeric ones.
11. `A>dir¦find "3-"¦find "-90"`
    Will search for all the files containing `3-` and `-90`, namely March, 1990.
13. `A>dir¦find"FILE.*"¦sort /+14`
15. `ECHO` sends `MORE`'s carriage return to the `DATE` command, thus causing both Date lines to be output automatically. `FIND` filters the line with C and redirects it to `DATEDATA`.

Chapter 9: Configuring the System

Drill

1.F; 3.a; 5.d; 7.T; 9.T; 11.F; 13.T; 15.T; 17.d; 19.F; 21.F; 23.F; 25.d; 27.a; 29.b.

Practices

9-1

1. `A>mem /debug`
3. `A>mode co40`
   `A>mod bw80`
   `A>mode co80`
5. `A>mode con`
   `A>mode lpt1`

9-2

1. `A>assign b=a`
   `A>dir`
3. `A>copy config.sys + con`
   `lastdrive=z`
   `^Z`
5. `A>join b: a:\xyz`

9-3

1. `A>break`
   `A>break on`
   `A>break off`
3. Include this in your `CONFIG.SYS` and reboot:
   `device=ansi.sys`

9-4

1. `A>append /e`

   `A>copy con addappnd.bat`
   `@echo off`
   `set append=%append%;%1`
   `^Z`
3. 1. Add a new line to `ENVPROM.BAT`:

   `A>copy envprom.bat + con`
   `set pc=Enter a command:`
   `^Z`

2. Add a new line to `SETPROM.BAT`:

   `A>copy setprom.bat + con`
   `if %1.==c. prompt=%pc%`
   `^Z`

3. Run `ENVPROM.BAT` and set the new prompt style with this:

`A>setprom c`

Review Questions

1. The /DEBUG switch causes the display of input/output devices while the /PROGRAM switch does not.
3. Keyboard (CON), monitor (CON), parallel printer (LPTn), serial printer (COMn), and modem (COMn).
5. 6/8 lines per inch and 80/132 columns.
7. Assigns B to A. DOS can no longer access B until cancellation or reassignment.
9. JOIN joins a real drive to an empty directory of another drive. The drive then becomes an extension of the directory. SUBST substitutes a real or logical (but nonexisting drive) for a directory of another drive. The directory need not be empty. The drive letter becomes a nickname for the directory.
11. It must be entered from a CONFIG.SYS file. It provides the number of disk buffers. A disk buffer is a block of memory used to store disk data.
13. An area of memory treated as a disk drive. Include this line in your CONFIG.SYS file:
```
device=c:\dos\ansi.sys
```
15. PATH, PROMPT, and APPEND (if installed with /E). Use the SET command to enter an environment variable, such as:
```
A>set var=value
```

Chapter 10: Using EDLIN

Drill

1.F; 3.T; 5.T; 7.T; 9.d; 11.d; 13.a; 15.d; 17.e; 19.T; 21.a; 23.T; 25.F; 27.T; 29.d.

Practices

10-1

1. In line 4, type up to `ready`, press Del twice, F3 once. In line 5, type up to `exa`, press Ins, type `ct`, press F3. In line 10, type up to `some`, press Del 13 times, press F3. In line 11, press Ins, type up to `fool`, press F1 until the end of `people`, press Del once, type `all`, press F3.

10-2

```
1. A>edlin saying
3. *1,6,12m
5. *q
   Abort edit (Y/N) y
```

10-3

```
1. A>edlin saying
   *1rFrancis ^Z
   *1rAbraham ^Z
3. *e
5. A>edlin saying
```

10-4

1.
```
A>edlin prez1
New file
*i
   1:*Nixon, Dick        309-4040
   2:*Ford, Jerry        123-3030
   3:*Carter, Jimmy      299-7777
   4:*Reagan, Ronny      333-6666
   5:*Bush, George       222-5555
   6:*^C
*e
```

3. `A>sort < prez > prez`

5. a. Create `PREZ1` in EDLIN.
   b. Create `PREZ` in EDLIN. After completion, use `T` to merge `PREZ1` to `PREZ`. Save `PREZ` and exit. Use `SORT` to sort `PREZ` and save over the original.

Review Questions

1. DOS cannot access EDLIN. There are a number of remedies. One is to make the directory where EDLIN is stored the current directory. Another is to provide a full path before EDLIN. Or you can use the `PATH` command to direct DOS to the drive or directory containing EDLIN.
3. EDLIN shows you `New file` or `End of input file`. The latter tells you that an old file has been retrieved and is ready for your action.
5. Deletes the whole file in the memory, from line 1 to the end.
7. The line is canceled and not registered. The old line, if any, remains.
9. F3.
11. `L` does not change the current line, but `P` does. With no parameters, `L` lists lines around the current line; `P`, however, goes down one screen each time.
13. EDLIN will start from after the current line and search for the previously entered string.
15. You are to precede that character with `^V` (Ctrl-V), such as `^V [`.

Chapter 11: Programming Batch Files

Drill

1.T; 3.F; 5.F; 7.c; 9.F; 11.F; 13.T; 15.b; 17.F; 19.F; 21.b; 23.a; 25.T; 27.T; 29.F.

Practices

11-1

1. `A>edlin demo.bat`
   Enter I and type the lines as shown. In line 4 press Ctrl-C. Enter E to exit.
3. Insert this in line 4:
   `pause`

5. Insert a line like this before and after each message line:

```
echo/
```

11-2

1.
```
1: @echo off
2: :AGAIN
3: echo %1
4: shift
5: if not .==%1. goto AGAIN
```

3.
```
1: @echo off
2: echo These files exist in source but not target:
3: for %%x in (%1) do if not exist %2:%%x echo %%x
4: echo Press a key to copy, or ^C to abort.
5: pause > nul
6: for %%x in (%1) do if not exist %2:%%x copy %%x %2:
```

11-3

1.
```
1: @echo off
2: rem LOOK.BAT--to look up strings in a file
3: if .==%1. goto HELP
4: if .==%2. goto HELP
5: find "%1" %2
6: goto END
7: :HELP
8: echo/
9: echo You forgot to enter two parameters, like this:
10: echo    %0 string filename
11: echo/
12: :END
```

3. 1. Change line 5 to this:

```
5: echo 0 = Italic 1 = NLQ 2 = Compressed 3 = Expanded
```

2. Insert these after line 7:

```
8: :0
9: echo  ^[4 > prn
10: goto END
```

3. Add ^[5 to the existing line 24:

```
24: echo ^[5^[x0^R^[W0^[P^[H > prn
```

11-4

1.
```
1: @echo off
2: rem FORMT.BAT--format and report outcome
3: c:\dos\format a:
4: if errorlevel 5 goto EL5
5: if errorlevel 4 goto EL4
6: if errorlevel 3 goto EL3
7: echo Formatting is successful
8: goto END
9: :EL3
```

```
10: echo Aborted due to pressing ^C
11: goto END
12: :EL4
13: echo Aborted due to an error
14: goto END
15: :EL5
16: echo Aborted due N repose for hard disk
17: :END
```

3. `A>if exist autoexec.bat echo found here`
   `C:\>if exist \dos\format.com echo It's here`

11-5

1. 1. Change `D - DBASE IV` to `E - EDLIN` in `MENU.TXT`.

   2. Change lines 9 and 12 to:

```
 9: if errorlevel 101 if not errorlevel 102 goto EDLIN
12: if errorlevel 69 if not errorlevel 70 goto EDLIN
```

   3. Change lines 18–19 to:

```
18: EDLIN
19: ed
```

   4. Create a batch file named `ED.BAT` with this line:

```
edlin test.bat
```

   (You may need a directory path showing where to access EDLIN and where to save your file.)

3.
```
 1: @echo off
 2: rem MENU.BAT--show available options
 3: :MENU
 4: cls
 5: echo Select: W(ordPerfect), L(otus), D(Base), Q(uit)
 6: key
 7: echo/
 8: if errorlevel 119 if not errorlevel 120 goto WP
 9: if errorlevel 113 if not errorlevel 114 goto END
10: if errorlevel 108 if not errorlevel 109 goto LOTUS
11: if errorlevel 100 if not errorlevel 101 goto DBASE
12: if errorlevel 87 if not errorlevel 88 goto WP
13: if errorlevel 81 if not errorlevel 82 goto END
14: if errorlevel 76 if not errorlevel 77 goto LOTUS
15: if errorlevel 68 if not errorlevel 69 goto DBASE
16: goto MENU
17: :WP
18: wp
19: :LOTUS
20: 123
21: :DBASE
22: db
23: :END
```

Review Questions

1. Put `@` before a command; put `@echo off` in the first line.
3. `Press any key to continue...` The following two lines will do it:
   ```
   echo Turn on the printer!!!!
   pause > nul
   ```
5. No. The topmost is always executed. DOS starts looking for a matching label from the top of a batch file. If one matching the spelling is found, all those below are ignored.
7. To assign the next parameter to the same dummy variable. If `%1` holds the first parameter the first time, after `SHIFT`, it holds the second parameter.
9.
   1. A file's existence—`IF EXIST FILENAME GOTO . . .`
   2. A parameter's existence—`IF .==%1. GOTO . . .`
   3. An exit code—`IF ERRORLEVEL 3 GOTO . . .`
11. `CALL` nests another batch file and returns execution to the calling batch file. Running batch 2 in the middle of batch 1 transfers control to batch 2. If batch 2 does not provide for going back to batch 1, other lines in batch 1, if any, are not executed.
13. They are responses to a command's various states of execution, such as successful formatting or certain error conditions. `BACKUP`, `FORMAT`, `REPLACE`, and `RESTORE` use exit codes.
15. It is a looping command. `FOR` specifies a variable, `IN` includes a list of files, and `DO` acts on the matching files.

Chapter 12: Using ANSI.SYS

Drill

1.F; 3.F; 5.T; 7.F; 9.d; 11.b; 13.c; 15.F; 17.c; 19.c; 21.a; 23.T; 25.F; 27.F; 29.T.

Practices

12-1

1.
```
A>prompt $e[1;1H$n$g$e[K
```
3.
```
A>prompt $e[2J
prompt
```
5.
```
1: @echo off
2: echo ^[[2J
3: echo ^[[12;34HHello World
4: prompt $e[1;25H
```

12-2

1.
```
1: @echo off
2: echo ^[[2J
3: echo ^[[32;43m
```

```
4: echo ^[[13;34HHello World
5: prompt $e[25;1H
6: echo ^[[0m
```

For mono, change line 3 values to `4;5`.

12-3

1. `A>prompt $e[27;"dir";13p`
3. `A>prompt $e[0;71;"dir\";13p`

12-4

1. `A>ansi 0;73;"dir..";13p`
3. `A>ansi 0;103;"dir|sort|more";13p`

12-5

1. Change `0;50` to `14` (`^N`), `0;33` to `6` (`^F`), `0;32` to `4` (`^D`), and `0;31` to `19` (`^S`). Also change instructions where appropriate.
3. `A>copy filename con`

12-6

1. Use 67 (F9), 92 (Shift-F9), and 102 (Ctrl-F9) to replace the numbers in the text.
3. 1. Define Ctrl-F9 as below, plus carriage return:

```
showmsg m0 m1 m2 m3 m4 m5 m6 m7 m8 m9
```

2. Create this batch file:

```
 1: @echo off
 2: rem SHOWMSG.BAT--show one message after another
 3: :AGAIN
 4: cls
 5: if .==%1. goto END
 6: if not exist %1 goto NOTEXIST
 7: echo/
 8: echo ---------------------- %1 -------------------
 9: echo/
10: type  %1
11: echo/
12: pause
13: shift
14: goto  AGAIN
15: :NOTEXIST
16: shift
17: goto  AGAIN
18: :END
19: echo/
20: echo That's all.
21: echo/
```

3. Press Ctrl-F9 to show one file at a time.

Review Questions

1. They are commands adopted by the American National Standards Institute. They are used to extend screen and keyboard functions. Each ANSI command (escape sequence) consists of Esc (ASCII 27), [(left bracket), a series of numbers, and ending with a letter.

3. `$e` is used with `PROMPT` exclusively. `^[` is entered in a file and then sent to the computer with `ECHO`, `COPY CON`, `TYPE`, or `MORE`.

5. The latest `PROMPT` command will go to the DOS environment, which may affect your future activities. An `ECHO` command does not go to the environment.

7. A `PROMPT` command has no effect until `ECHO` is turned on again. A command such as `prompt $e[2J` will erase every screen display when `ECHO` is on. Its feature is disabled when `ECHO` is off.

9. 1. Set monitor to high intensity, blink.
   2. Set to default attributes; `m` alone is same as `0m`.

11. Shows blinking message preceded and followed by a blank line. Blinking is turned on and off by the first and last lines.

13. Use `TYPE`, `COPY`, or `MORE` to send the file to the screen, such as:
    ```
    A>type filename
    ```

15. Assigns to Alt-D the string inside the quotes, plus a carriage return. Pressing Alt-D then appends two-line date data to a file named `NOTES`.

Index

. 104, 129, 294
.. 129, 135, 136
: 319, 323, 331
? 99, 101, 105, 306–308
$ 139
$$$ 311
$e[335–338
-/+ 291, 301, 304
+ 94, 158–163, 166–168
+/- A 193–194
+/- R 193–194
* 99, 102–104, 288
^@ 309
^C 38
^Z 93, 107, 166–168, 210, 307, 311
^[[335–338
< 209, 223, 225, 229
= 276–280, 255, 257, 261–262
== 279, 319, 332
> 219, 220–222, 229, 244
>> 219, 220, 229, 243–245
% 112, 242, 319, 324–326
%% 242, 327–328
%var% 277–280, 335
292, 295
@ 109, 319, 321
\ 108, 128, 135–136
¦ 104, 219, 227, 229, 243–245
/ 39
→ 168

A

Aborting a command. *See* Ctrl-C
Abort, Retry, Ignore, Fail? 40
Allocation unit 77
Alphanumeric characters 236, 358
ANSI escape sequence 355–356
ANSI.SYS 270, 355–356
APPEND 146–149, 275–276
Appending lines 94, 244–245
Appending output 220, 222, 229, 244–245
Application software 15
Archive 193
Argument 37. *See also* Replaceable parameter
Arrow keys 107
ASCII 386, 400, 442
ASCII files 32–33
ASSIGN 261–262
AT 3–5
ATTRIB 193–194
Attributes 77, 193
AUTOEXEC.BAT 113–115, 124
AUX 98

B

BACKUP 198–205
BACKUP.nnn 200
Backspace key 107
BAK file extension 293
BAT file extension 33
Batch commands 319
Batch files 33, 110–116
  A.BAT/B.BAT 343
  A2B.BAT 326
  ADDPATH.BAT 280
  ANSI.BAT 372–373
  AUTOBACK.BAT 204, 339
  BRKTEST.BAT 269
  BACK.BAT 338
  DATEFILE.BAT 244
  DELALL.BAT 224
  DIARY.BAT 245
  DUPE.BAT 337
  E.BAT 312
  EPSON.BAT 333
  ENVPROM.BAT 279
  FLAG.BAT 362
  FMAT.BAT 225
  KILL.BAT 330–331
  MEMO.BAT 374–325
  MEMO1.BAT 376–77
  MENU.BAT 344–348
  MOVE.BAT 333

Batch files *(continued)*
MSG.BAT 378
PARADEMO.BAT 326
PASS.BAT 335
RAMCOPY.BAT 273
SEEFILE.BAT 241
SEELINE.BAT 242
SEEMORE.BAT 230
SETPROM.BAT 279
SHOWBACK.BAT 328
SHOWTIME.BAT 361
speedy creation 116, 312
TESTFILE.BAT 422
TESTPROM.BAT 277
Baud 11, 260
Binary 441–442
Binary files 33, 311
BIOS 8
Bit 441
Blank character 100, 335
Boot 8, 17–18
Boot disk 73
Boot record 76
Branching 330
BREAK 268–269
Break key. *See* Ctrl-Break
BUFFERS 268–269
buffer 308. *See also* Template
Bug 22, 320, 401
Bus 5
Byte 442

C

Calendar 24
CALL 319, 341–344
Canceling a command. *See* Esc
CD (CHDIR) 129, 316
CD-ROM 63
Central processing unit. *See* CPU
CGA 9
Chaining 342–344
Changing a file name. *See* REN
Changing a file's date 161–162
CHCP 404
CHKDSK (Check disk) 80–83
Clones 4
CLS (Clear the screen) 22
Cluster 77–78
Code page 259, 404
Cold boot 18
COM1 98, 259–260
Combined keys 14
Combining files 157–163
COMMAND 402
Command 21
COMMAND /C 343
COMMAND.COM 73, 124, 188
Command line 21
Command line argument 37
COMP (Compare files) 177–179
Comparing disks. *See* DISKCOMP
Comparing files 177–180
Compatibles 4
Computer radiation 25
Computer virus 195
COMSPEC 275
CON (Console) 96, 98, 167–168, 255
Concatenating files 157–163
CONFIG.SYS 124, 263, 268
Configuration commands 267–270
Content of destination lost... 160
Control characters 309–310
CONTROL.nnn 200
Controller card 8
COPY
combining files 158–161, 162–163
duplicating files 44–46, 176, 257, 328
printing files 55
stamping current date 161–163
to devices 167–168
COPY CON
adding to files 94
creating files 93
difference with TYPE 105, 168
Copy-protected files 176–177
CP/M 15
CPU 5–7
8088/8086 5
80286 6
80386 6
80486 7
Creating a directory 128
Creating a file 93
Ctrl-Alt-Del 14, 18
Ctrl-Break 14, 38
Ctrl-C 14, 38, 258, 292
Ctrl-Num Lock 14
Ctrl-P (Ctrl-PrtSc) 14
Ctrl-S 14
Ctrl-V 309–310, 356
Ctrl-Z 93
CTTY 402
Current directory 123
Current line 288, 292
Cursor 21, 359–361
Cursor keypad 12–13
Cylinders 85

D

Daisy-wheel printer 10
DAT 83
DATE 22–24, 243
DEBUG 345
Debugging 320
Default drive 34
DEL (Delete files) 48–49, 103–104, 131
Del (Delete) key 107
DEVICE 268
Device drivers 270
Device names
AUX 98
COM1 98, 259–260
CON 98, 167–168, 255
LPT1 55, 98, 257–259
NUL 98, 221, 321
PRN 55, 98, 167–168
DIR 35–38, 103, 129, 235
Directory 77, 123
Directory entries 35–36
Disk
boot record 76
buffers 269
capacities 68–70
components 64–65
drives 8
FAT 76–77
fixed 63
floppy 64
fragmentation 82–83
hard 63, 85–86
hub 64
index hole 64–65
organization 76–78
partition 85
read-write opening 64–65
sector 68
sector notch 64–65
spindle 65
track 65
write-protect notch 64–65
DISKCOMP (Disk compare) 180–181
DISKCOPY 50–52, 174–176
DISPLAY.SYS 404
DOC file extension 32

DOS
commands 21
device names. *See* Device names
editing keys 106–110
environment 147, 274–280, 335
prompt 20–21
special keys 13–14
symbols. *See* the beginning of this index
versions 22, 139
DOS.EXE xv
DOSLESnn xv
DOSSHELL.BAT 435
Dot-matrix printer 10
DRILL.EXE xv
DRIVER.SYS 404
Dummy variables 112, 324

E

ECHO 104, 222, 319, 320–322, 356–359
Echo from screen to printer 14
Editing keys 106–110
EDLIN
append 311
backup file 293
binary files 311
buffer 298
commands 291
copy 299–302
current line 288, 292
delete 301
edit 294
editing keys 295
end of file 292, 295
exit and save 289
insert 290
list 290, 303–305
move 299–300
page 305
prompt 288
quit 302
replace 307–308
search 306–307
transfer 311
write 311
EGA 9
EISA 5
End-of-file character. *See* ^Z
Enhanced keyboard 13
Environment 147, 274–280
Environment variable 276–280, 335
ERASE. *See* DEL
Errorlevel 338–340, 346
Esc character 355–356
Esc key 14, 21, 107–108
EXE file extension 33
Executable files 33
EXIST 319, 337–340
EXIT 402
Exit codes 197, 338–340
Expanded memory 270, 390
Expansion slots 8
Extended key codes 371
Extended memory 253, 390
External commands 52–53

F

FASTOPEN 404
FAT (File allocation table) 76–77, 84
FC (File compare) 179–180
FCBS 403
FDISK 85
File 32–33
attributes 77, 193
combination 157–163
contiguity 82–83
creation 93
deletion. *See* DEL
handles 270
inserting date-time 243–246
names 96–99
renaming 47, 103
stamping current date 161–162
types 32–33
FILEnnnn.CHK 83
FILEnnnn.REC 84
FILES 268, 270
Filter 228, 241
FIND 238–241
Firmware 8
Fixed disk 63, 85–86
Floppy disks 64
FOR..IN..DO 242, 319, 327–328, 379
FORMAT 49, 86, 199
Function keys 12–13, 107, 369–371

G

Gigabyte 442
GOTO 319, 330–331
GRAFTABL 304
GRAPHICS 54

H

Halting a screen display 41
Hard disk 63, 85–86, 124
Hercules 9
Hexadecimal 442
Hidden files 187, 189

I

IBM 3–5
IBMBIO.COM 73, 188
IBMDOS.COM 73, 188
IF 319, 330–336
IF EXIST 337–340
IF NOT 319, 333–336
IF NOT EXIST 337–340
Illegal characters 97
Index hole 64–65
Input redirection. *See* Redirecting input
Ins (Insert) key 107
Inserting Current date 243–246
Intel 5
Internal commands 52–53
IO.SYS 73, 188

J

JOIN 264–265

K

KB (Kilobyte) 3, 442
Key codes 369–371
KEYB 100, 405
KEYBxx 403
Keyboard 13
KEYBOARD.SYS 100
KEY.COM xvi, 345

L

LABEL 79–80
Label in batch file 330–331
Laser printer 10
LASTDRIVE 262
Line editor 287
Logical drive 262
Look-ahead 270
Looping 327–328
LPT1 55, 98, 257–259

M

Macro 355
Mainframe 3
MB (megabyte) 442
MCA 5
MD (MKDIR) 128

MDA 9
MEM 253–254
Memory-resident. *See* TSR
Menu building 344–348
MENUB.BAT xvi
MENU.EXE vxi
Metastring 139
Microchannel 5
Microcomputer 3
Minicomputer 3
MODE 254–260
  and device status 258–260
  and keyboard 254–255
  and monitors 256–257
  and printers 257–258
Modem 11, 260
Monitors 9
Monochrome 9
MORE 228–232
Motherboard 8
Mouse 11
MS-DOS 15, 73, 188–189
MSDOS.SYS 73, 188
Multitasking 16, 209

N

Nesting 341–344
NLQ (near-letter quality) 10
NLSFUNC 404
NUL 98, 221, 321
Numeric keypad 12, 97–98, 362

O

Octal 442
Output redirection. *See* Redirecting output
Operating system 15
Operating environment 16
OS/2 15–16

P

Parallel port 9
Parameter 37
Parameter testing 332
Parent directory 124
Partition 85
Password 334–336
PATH 145–146, 275–276, 280
Path 43, 280
Path name 43
PAUSE 319, 320–322
Pause key 14
PC 3–5
PC-DOS 15, 73, 188–189
Phone message 377–380
Pipe 219, 227
Pixels 9
Platter 85
Port 9–10
PRINT 209–212
Print queue 210
Print Scrn 14, 53–54
PRINTER.SYS 404
Printers 9–10
  and MODE 257–258
Printing
  a line with ECHO 222
  and Displaying 54–55
  disk files 55
  in the background 209–212
  redirected output 55–56
  screen text 53
  screen graphics 54
  using COPY 55, 167–168
  using MORE 231–232
PRN 55, 98, 167–168
PROMPT 138–141, 275–279, 356–8, 374
PS/1-2 4

R

RAM 3, 8, 253–254
RAM disk 271–273
RAMDRIVE.SYS 271
RD (RMDIR) 127, 130
README 32
Read-only 193
Read-write opening 64–65
Rebooting 17–18
RECOVER 84
Redirecting input 219, 223–225, 244
Redirecting output 219–223, 244
Redirection 219
REM (Remark) 319, 323
REN (Rename files) 47, 103
REPLACE 190–192
Replaceable parameter 112, 230, 241–2, 273, 324–6
Rerouting drives 261–266
Reserved words 98
Reset button 17
RESTORE 206–209
RGB 9
ROM 8
Root directory 77, 123–124

S

Scanner 11
Sector 68
Sector notch 64–65
SELECT 403
Serial port 10, 260
SET 269, 274–277, 335
SHARE 403
SHELL 275, 404–405
Shell 435
SHIFT 319, 324–326
Shift-Print Scrn 14, 53–54
Simultaneous displaying and printing 54–55
SORT 233–237
Source 45, 190
Stamping current date 161–162
Standard input 219
Standard output 219
Subdirectory 124
SUBST 263–264
Surge protector 17
Switches 39
Symbols. *See* the beginning of this index
Syntax 45
SYS 187–188
SYS file extension 33
System
  disk 19, 187–189
  prompt 20–21
  reset 17
  software 15
  unit 8

T

Target 45, 190
Template 106
Text editor 287
TIME 24, 24, 243
Toggle 12
Track 68
TREE 143
TSR (Terminate and stay resident) 149, 212
TXT file extension 32
TYPE 42, 229–230, 328

U

UNIX 15–16
Updating files 161–162

V

Variables 324, 327. *See also* Replaceable parameter and Environment variable
VDISK.SYS 271
VER (Version) 22
VERIFY 165–166
VGA 9
Video graphics card 9
Virtual disk 271–273
VOL 79
Volume label 79

W

Warm boot 18
Wild card characters 104–105
Windows 16
Word processor 15, 287
Write-protect notch/switch 64–65

X

XCOPY 170–173, 176, 194–197
XT 3–4
XMAEM.SYS 405
XMA2EMS.SYS 405

The DOS Coursebook

by Forest Lin

COMBINES POWERFUL PEDAGOGY AND THOROUGH COVERAGE:

Covers powerful but rarely explained commands like FOR and SET; manipulating the DOS environment; batch file programming techniques (such as looping and variable handling); ANSI color/cursor manipulations.

Includes Drill Questions after each section, Answers to the odd-numbered questions, Practice Exercises for "hands on" work, and Review Questions.

COVERAGE OF BOTH VERSIONS 3.3 AND 4.0

SPECIAL SECTIONS FOR "HARD DISK USERS"

FREQUENT "TIPS AND TRICKS"

Covers common errors made by beginning students, and advanced techniques which "power users" will appreciate!

THE DOS-TA (DOS TEACHING ASSISTANT) DISK

Includes an online help program for DOS commands; a CAI program; a practice text file used in the book...and more!

ABOUT THE AUTHOR

Forest Lin teaches at Tulsa Junior College. He has taught computer-related topics and courses for 20 years, and is the author of three other computer-education texts.

ISBN 0-9624230-3-3

Scott/Jones, Inc.
P.O. Box 696
El Granada, CA 94018
415-726-2436 or 726-7090